EXHIBIT P

PROVISIONAL APPLICATION FOR PATENT COVER SHEET

This is a request for filing a PROVISIONAL APPLICATION FOR PATENT under 37 CFR 1.53(c).

Express Mail Label No. ____________

INVENTOR(S)

Given Name (first and middle [if any])	Family Name or Surname	Residence (City and either State or Foreign Country)
Andrew Everett	Phelps	Middleton, WI
Norman Paul	Jouppi	Palo Alto, CA

Additional inventors are being named on the ____________ *separately numbered sheets attached hereto*

TITLE OF THE INVENTION (500 characters max):

PERFORMING MATRIX MULTIPLICATION IN HARDWARE

Direct all correspondence to: **CORRESPONDENCE ADDRESS**

☒ The address corresponding to Customer Number: 26192

OR

☐ Firm or Individual Name

Address		
City	State	Zip
Country	Telephone	Email

ENCLOSED APPLICATION PARTS *(check all that apply)*

☒ Application Data Sheet. See 37 CFR 1.76

☐ CD(s), Number of CDs ____________

☒ Drawing(s) *Number of Sheets* 13

☐ Other (specify) ____________

☒ Specification (e.g. description of the invention) *Number of Pages* 39

Fees Due: Filing Fee of $260 ($130 for small entity) ($65 for micro entity). If the specification and drawings exceed 100 sheets of paper, an application size fee is also due, which is $400 ($200 for small entity) ($100 for micro entity) for each additional 50 sheets or fraction thereof. See 35 U.S.C. 41(a)(1)(G) and 37 CFR 1.16(s).

METHOD OF PAYMENT OF THE FILING FEE AND APPLICATION SIZE FEE FOR THIS PROVISIONAL APPLICATION FOR PATENT

☐ Applicant claims small entity status. See 37 CFR 1.27.

☐ Applicant certifies micro entity status. See 37 CFR 1.29.
Applicant must attach form PTO/SB/15A or B or equivalent.

☐ A check or money order made payable to the *Director of the United States Patent and Trademark Office* is enclosed to cover the filing fee and application size fee (if applicable).

☐ Payment by credit card. Form PTO-2038 is attached

☒ The Director is hereby authorized to charge the filing fee and application size fee (if applicable) or credit any overpayment to Deposit Account Number: 06-1050.

$260

TOTAL FEE AMOUNT ($)

The invention was made by an agency of the United States Government or under a contract with an agency of the United States Government.

☒ No

☐ Yes, the invention was made by an agency of the U.S. Government. The U.S. Government agency name is: ______

☐ Yes, the invention was made under a contract with an agency of the U.S. Government. The name of the U.S. Government agency and Government contract number are: ______

SIGNATURE /Stephanie D. Grosvenor/ Date May 5, 2017

TYPED or PRINTED NAME Stephanie Grosvenor REGISTRATION NO. 67,592 *(if appropriate)*

TELEPHONE (650) 839-5143 Docket Number: 16113-8252P01

51055044.doc

PERFORMING MATRIX MULTIPLICATION IN HARDWARE

BACKGROUND

This specification relates to performing neural network computations in hardware.

Neural networks are machine learning models that employ one or more layers to generate an output, e.g., a classification, for a received input. Some neural networks include one or more hidden layers in addition to an output layer. The output of each hidden layer is used as input to another layer in the network, e.g., the next hidden layer or the output layer of the network. Each layer of the network generates an output from a received input in accordance with current values of a respective set of parameters.

SUMMARY

In general, this specification describes a special-purpose hardware circuit that computes neural network inferences.

In general, one innovative aspect of the subject matter described in this specification can be embodied in methods of performing a matrix multiplication using a hardware circuit that include the actions of obtaining, by a matrix computation unit of the hardware circuit, an input activation value and a weight input value, the input activation value and the weight input value each having a first floating point format; multiplying, by a multiplication circuitry of the matrix computation unit, the input activation value and the weight input value to generate a product value, the product value having a second floating point format that has a higher precision than the first floating point format and has a dynamic range that is at least as large as the dynamic range of the first floating point format; obtaining, by the matrix computation unit, a partial sum value in a third floating point format that has a higher precision than the first floating point format and has a dynamic range that is at least as large as the dynamic range of the first floating point format; and combining, by a summation circuitry of the hardware circuit, at least the partial sum value and the product value to generate an updated partial sum value that has the third floating point format.

Embodiments of this aspect can include one or more of the following optional features. The precision of a floating point format can be determined based on a count of available bits for a significand in the floating point format and the dynamic range of a floating point format can be determined based on a count of available bits for an exponent in the floating point format. The second floating point format can have the same dynamic range as the first floating point format and the third floating point format can have the same dynamic range as the first floating point format. The third floating point format can have a higher precision than the second floating point format.

The hardware circuit can be configured to perform computations for a neural network having a plurality of layers, and the input activation value and the weight input value can be associated with a layer of the plurality of layers.

The methods can include the actions of obtaining a raw activation value and a raw weight value for the first matrix computation cell having the third floating point format; converting the raw activation value into the first floating point format to generate the input activation value; and converting the raw weight value into the first floating point format to generate the weight input value. The methods can further include the actions of receiving a request to process the raw activation value with enhanced precision; generating an activation enhanced precision value for the input value, the activation enhanced precision value being a measure of difference between the activation input value and the raw activation value; and generating a weight enhanced precision value for the weight input value, the weight enhanced precision value being a measure of difference between the weight input value and the raw weight value. Multiplying the activation input value and the weight input value to generate the product value can include the actions of multiplying, by the multiplication circuitry of the matrix computation unit, the input value by the weight input value, the input value by the weight enhanced precision value, the weight input value by the activation enhanced precision value, and the activation enhanced precision value by the weight enhanced precision value; and combining products of the multiplications to generate the product value.

Other embodiments of this aspect include corresponding computer systems, apparatus, and computer programs recorded on one or more computer storage devices, each configured to perform the actions of the methods. A system of one or more computers can be configured to perform particular operations or actions by virtue of software, firmware, hardware, or any combination thereof installed on the system that in operation may cause the system to perform the actions. One or more computer programs can be configured to perform particular operations or actions by virtue of including instructions that, when executed by data processing apparatus, cause the apparatus to perform the actions.

Another innovative aspect of the subject matter described in this specification can be embodied in hardware circuits that include a matrix computation unit configured to perform a first group of operations including: obtaining an activation input value and a weight input value, the activation input value and the weight input value both having a first floating point format; storing the weight input value in a weight register, the weight register being configured to store values having the first floating point format; multiplying, a using multiplication circuitry of the hardware circuit, the activation input value and the weight input value to generate a product value, the product value having a second floating point format that has a higher precision than the first floating point format and has a dynamic range that is at least as large as the dynamic range of the first floating point format; obtaining a partial sum value in a third floating point format that has a higher precision than the first floating point format and has a dynamic range that is at least as large as the dynamic range of the first floating point format; storing the partial sum value in a sum in register, the sum in register being configured to store values having the third floating point format; and combining, using a summation circuitry of the matrix computation unit, the partial sum value and the product value to generate an updated partial sum value that has the third floating point format.

Embodiments of this aspect can include one or more of the following optional features. The first group of operations can include storing the activation input value in an activation register, the activation register being configured to store values having the first floating point format. The first group of operations can include storing the weight input value in a weight register, the weight register being configured to store values having the first floating point format. The first group of operations can include storing the partial sum value in a sum in register, the sum in register being configured to store values having the third floating point format. The hardware circuit can include an external summation circuitry outside the matrix computation unit. The first group of operations can include receiving a request to process the raw activation value with enhanced precision; generating an activation enhanced precision value for the input value, the activation enhanced precision value being a measure of difference between the activation input value and the raw activation value; and generating a weight enhanced precision value for the weight input value, the weight enhanced precision value being a measure of difference between the weight input value and the raw weight value. Multiplying the activation input value and the weight input value to generate the product value can include the actions of multiplying, by the multiplication circuitry of the matrix computation unit, the input value by the weight input value, the input value by the weight enhanced precision value, the weight input value by the activation enhanced precision value, and the activation enhanced precision value by the weight enhanced precision value. The external summation circuitry can be configured to perform a second group of operations including combining products of the multiplications to generate the product value.

Other embodiments of this aspect include corresponding computer systems, apparatus, and computer programs recorded on one or more computer storage devices, each configured to perform the actions of the first group of operations and/or the second group of operations. A system of one or more computers can be configured to perform particular operations or actions by virtue of software, firmware, hardware, or any combination thereof installed on the system that in operation may cause the system to perform the actions. One or more computer programs can be configured to perform particular operations or actions by virtue of including instructions that, when executed by data processing apparatus, cause the apparatus to perform the actions.

Particular embodiments of the subject matter described in this specification can be implemented so as to realize one or more of the following advantages. A hardware circuit can perform matrix multiplication with reduced overflow and/or loss of precision. A hardware circuit can perform matrix multiplication with enhanced precision beyond the precision provided by the floating point format of input registers in the hardware circuit. A hardware circuit can perform matrix multiplication on an input matrix whose values are stored using an IEEE single-precision floating point format with reduced overflow even though the hardware circuit stores the input matrix values in a floating point format with 16 bits.

The details of one or more embodiments of the subject matter of this specification are set forth in the accompanying drawings and the description below. Other features, aspects, and advantages of the subject matter will become apparent from the description, the drawings, and the claims.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1A shows a high-level diagram of an example special-purpose hardware chip for training a neural network.

FIG. 1B shows a high-level example of compute core.

FIG. 1C shows an example neural network processing system.

FIG. 2 illustrates an example architecture including a matrix multiply unit. The matrix multiply unit is a two-dimensional systolic array.

FIG. 3 illustrates an example architecture of a multi-cell inside a systolic array.

FIGS. 4A-4B show example architectures that each include a matrix computation unit.

FIG. 5 shows an example architecture of a cell in a matrix computation unit.

FIG. 6 shows an example architecture of a vector computation unit.

FIG. 7 shows an example format for a floating point value.

FIG. 8 shows an example architecture for multiplication circuitry of a matrix computation cell.

FIG. 9 is a flow diagram of an example process for performing matrix multiplication using a matrix computation unit.

FIG. 10 is a flow diagram of an example process for performing multiplication of an activation input value by a weight input value.

Like reference numbers and designations in the various drawings indicate like elements.

DETAILED DESCRIPTION

A neural network having multiple layers can be used to perform computations. For example, given an input, the neural network can compute an inference for the input. The neural network computes this inference by processing the input through each of the layers of the neural network. Each layer receives an input and processes the input in accordance with the set of weights for the layer to generate an output.

Therefore, in order to compute an inference from a received input, the neural network receives the input and processes it through each of the neural network layers to generate the inference, with the output from one neural network layer being provided as input to the next neural network layer. Data inputs to a neural network layer, e.g., either the input to the neural network or the outputs of the layer below the layer in the sequence, to a neural network layer can be referred to as activation inputs to the layer.

In some implementations, the layers of the neural network are arranged in a sequence. In some other implementations, the layers are arranged as directed graph. That is, any particular layer can receive multiple inputs, multiple outputs, or both. The layers of the neural network can also be arranged such that an output of a layer can be sent back as an input to a previous layer.

The neural network can also be trained to determine trained values of the weights of the layers in the neural network. Generally, during the training, inputs are processed using the neural network and the weights are adjusted based on the outputs generated by the neural network for the inputs.

This specification describes special-purpose hardware circuitry that performs neural network computations, i.e., inference or training operations, including matrix multiplication operations performed by the neural network layers.

FIG. 1A shows a high-level diagram of an example special-purpose hardware chip for training a neural network. As illustrated, a single special-purpose hardware chip includes two independent processors (102a, 102b). Each processor (102a, 102b) contains two distinct cores: (1) a compute core, i.e. a very long instruction word (VLIW) machine, (103a, 103b) and (2) a sparse computation core, i.e. an embedding layer accelerator, (105a, 105b).

Each compute core (103a, b) is optimized for dense linear algebra problems. A single, very long instruction word controls several compute cores in parallel. Each compute core has three types of compute units.

An example sparse computation core (105a, b) maps very sparse, high-dimensional data into dense, low-dimensional data so that the rest of the layers process densely packed input data. For example, the sparse computation core can perform the computation of any embedding layers in the neural network being trained.

To perform this sparse-to-dense mapping, the sparse computation core uses a pre-built lookup table, an embedding table. For example, when there is a series of query words as user input, each query word is converted into a hash identifier or a one-hot encoded vector. Using the identifier as a table index, the embedding table returns the corresponding dense vector, which can be an input activation vector to the next layer. The sparse computation core can also perform reduction operations across the search query words to create one dense activation vector. The sparse computation core performs efficient sparse, distributed lookups since the embedding table can be huge and not fit in the limited capacity high bandwidth memory of one of the special-purpose hardware chips. More details about the sparse computation core functionality can be found in U.S. Patent Application No. 15/016,486, entitled MATRIX PROCESSING APPARATUS, which was filed on February 5, 2016.

FIG. 1B shows a high-level example of compute core (101). The compute core can be a machine, i.e. a VLIW machine, that controls several compute units in parallel. Each compute core (101) contains: a scalar memory (104), a vector memory (108), a scalar processor (107), a vector processor (106), and extended vector units (i.e. a matrix multiply unit (MXU) (113) a transpose unit (XU) (114), and a reduction and permutation unit (RPU) (116)).

An example scalar processor performs VLIW instruction fetch/execute loop and controls the compute core. After fetching and decoding an instruction bundle, the scalar processor itself only executes the instructions found in the scalar slots of the bundle using multiple, multi-bit registers, i.e. 32 32-bit registers of the scalar processor (107) and scalar memory (104). The scalar instruction set includes normal arithmetic operations, e.g., as used in address calculations, load/store instructions, and branch instructions. The remaining instruction slots encode instructions for the vector processor (106) or other extended vector units (113, 114, 116). The decoded vector instructions are forwarded to the vector processor (106).

Along with vector instructions, the scalar processor (107) can forward values of up to three scalar registers to the other processor and units to perform operations. The scalar processor can also directly retrieve computation results from the vector processor. However, in some implementations, the example chip has a low-bandwidth communication path from the vector processor to the scalar processor.

A vector instruction dispatcher sits between the scalar processor and the vector processor. This dispatcher receives decoded instructions from the non-scalar VLIW slots and broadcasts those instructions to the vector processor (106). The vector processor (106) is described in detail with respect to FIG. 1C.

An example scalar processor (107) accesses a small, fast, private scalar memory (104), which is backed up by a much larger, but slower High Bandwidth memory (HBM) (110). Similarly, an example vector processor (106) accesses a small, fast, private vector memory (108), which is also backed up by the HBM (110). Word-granularity access occurs between either the scalar processor (107) and the scalar memory (104) or the vector processor (106) and the vector memory (108). Direct memory access occurs between the scalar memory (104) and the HBM (110) and the vector memory (108) and the HBM (110). In some implementations, memory transfers from the HBM (110) to the processors (107, 106) may only be done through the scalar or vector memories. Additionally, there may be no direct memory transfers between the scalar memory and the vector memory.

Instructions may specify extended vector unit operations. Along with each executed vector unit instruction, there are two-dimensional, i.e. 128 by 8, vector units that each can send one register value to the extended vector units as input operands. Each extended vector unit takes the input operands, performs corresponding operations, and returns the results back to the vector processor (306). The extended vector units will be described below with respect to FIG.4.

FIG. 1C shows an example special-purpose integrated circuit 100 for performing neural network computations. As illustrated, the chip contains two compute cores (103a, 103b) and two sparse computation cores (152a, 152b).

The chip has a shared area which includes a host interface to a host computer (150), four stacks of high bandwidth memory along the bottom (156a-156d), and an inter-chip interconnect (148) connecting the interfaces and memory together, as well as data from other chips. Two stacks of high bandwidth memory (156a-b, 156c-d) are associated with each compute core (103a, 103b).

The chip stores data in high bandwidth memory (156c-d), reads the data in and out of vector memory (108), and processes the data. The compute core (103b) itself includes a vector memory (108) that is on-chip S-RAM which is divided into two dimensions. The vector memory has address space in which addresses hold floating point numbers, i.e. 128 numbers that are each 32-bits. The compute core (103b) also includes a computational unit that computes values and a scalar unit that controls the computational unit.

The vector processor (106) consists of a 2-dimensional array of vector processing units, i.e. 128 x 8, which all execute the same instruction in a single instruction, multiple-data (SIMD) manner. The vector has lanes and sublanes, i.e. 128 lanes and 8 sublanes. In each lane, there are N vector units across the sublane dimension. These N vector units share an N-banked vector memory that is private to each lane. Within the lane, the vector units communicate with each other through load and store instructions. Each vector unit can access one 4-byte value at a time. Vector units that do not belong to the same lane cannot communicate directly. These vector units must use the reduction/permutation unit which is described below.

The computational unit includes vector registers, i.e. 32 vector registers, in a vector processing unit (106) that can be used for both floating point operations and integer operations. The computational unit includes two arithmetic logic units (ALUs) (106c-d) to perform computations. One ALU (106c) performs floating point addition and the other ALU (106d) performs floating point multiplication. Both ALUs (106c-d) can perform various other operations such as shifts, masks, and compares. For example, a compute core (103b) may want to add a vector register, V1, and a second vector register, V2, and put the results in a third vector register, V3. In order to compute the addition, the compute core (103b) performs multiple, i.e. 1024, operations in one clock cycle. Using these registers as operands, each of the vector units can simultaneously execute two ALU instructions, one load and one store instruction, every clock cycle. A base address for a load or a store instruction can be computed in the scalar processor and forwarded to the vector processor. Each of the vector units in each sublane can compute its own offset address using various methods such as striding and a special indexed address register.

The computational unit also contains an extended unary pipeline (EUP) (116) that performs operations such as square root and reciprocal. The compute core (103b) takes three clock cycles to perform these operations since they take in one operand at a time. Since the EUP processing takes more than one clock cycle, there is a first-in-first-out data storage to store results. When an operation is finished, the results are stored in the FIFO. The compute core can use a separate instruction at a later time to pull the data out of the FIFO and put it in the vector register. A random number generator (120) allows the compute core (103b) to generate random numbers per cycle, i.e. 128 random numbers per cycle.

As described above, each processor has three extended vector units: a matrix multiply unit (138) which performs matrix multiplication operations; a transpose unit (122) which performs a transposition operation of a matrix, i.e. 128 by 128 matrix, and a reduction and permutation unit (illustrated as separate units in FIG. 4 124, 126).

The matrix multiply unit performs matrix multiplications between two matrices. The matrix multiply unit (138) takes in data since the compute core needs to load in a set of numbers which is the matrix that is going to be multiplied. As illustrated, data comes from the vector registers (106). Each vector register contains a number, i.e. a 32-bit number. However, floating point conversion may occur as data is sent to the matrix multiply unit (138) to change the numbers to a smaller bit size, i.e. from 32-bit to 16-bit. A serializer (130) ensures when numbers are read out of the vector registers, a two-dimensional array, i.e. a 128 by 8 matrix, is read as sets of 128 numbers that are sent to the matrix multiply unit (138) for each of the next eight clock cycles. After the matrix multiply has completed its computations, the results are deserialized (132a,b) which means that result matrix is held for a number of clock cycles. For example, for a 128 x 8 array, 128 numbers are held for each of 8 clock cycles and then pushed to a FIFO so that a two-dimensional array of 128 X 8 numbers can be grabbed in one clock cycle and stored in the vector registers (106).

Over a period of cycles, i.e. 128 cycles, weights are shifted into the matrix multiply unit (138) as the numbers by which to multiply the matrix. Once the matrix and weights have been loaded, the compute core (103b) can send sets of numbers, i.e. 128 X 8 numbers, to the matrix multiply unit (138). Each line of the set can be multiplied by the matrix to produce a number of results, i.e.128, results per clock cycle. While the compute core is performing matrix multiplies, the compute core also shifts new sets of numbers in the background to be the next matrix by which the compute core will multiple so that the next matrix is available when the computational process for the previous matrix has completed. The matrix multiply unit (138) can process weight inputs and activation inputs and provide a vector of outputs to the vector processing unit 106. The vector processing unit can process the vector of outputs and store a vector of processed outputs to the vector memory. For example, the vector processing unit can apply a non-linear function to outputs of the matrix multiply unit to generate activated values. In some implementations, the vector computation unit 114 generates normalized values, pooled values, or both. The vector of processed outputs can be used as activation inputs to the matrix multiply unit 112, e.g., for use in a subsequent layer in the neural network.

The transpose unit transpose a matrix. The transpose unit (122) takes in numbers and transposes them so that the number across a lane is transposed with the number in the other dimension. In some implementations, the vector processor includes 128 x 8 vector units. Therefore, to transpose a 128 x 128 matrix, sixteen individual transpose instructions are needed for the full matrix transpose. Once the transposition is finished, the transposed matrix will be available. However, an explicit instruction is needed to move the transposed matrix into the vector register file.

The reduction/permutation unit (or units 124, 126) addresses the problem of cross-lane communication by supporting various operations such as permutation, lane rotation, rotating permutation, lane reduction, permuted lane reduction, and segmented permuted lane reduction. As illustrated, these computations are separate, however, a compute core can use one or the other or one chained to the other. The reduction unit (124) reduces each line of numbers and feeds the numbers into the permutation unit (126). The permutation unit alters data between different lanes. The transpose unit, the reduction unit, the permutation unit, and the matrix multiply unit each take more than one clock cycle to complete. Therefore, each unit has a FIFO associated with it so that the results of computations can be pushed to the FIFO and a separate instruction can be executed at a later time to pull the data out of the FIFO and into a vector register. By using FIFOs, the compute core does not require multiple vector registers to be reserved for the duration of lengthy operations. As illustrated, each of the units takes data from the vector registers (106).

The compute core uses a scalar unit to control the computational unit. The scalar unit has two primary functions: (1) performing loop counting and addressing and (2) generating direct memory address (DMA) requests so that the DMA controller moves data in the background between the high bandwidth memory (156c-d) and vector memory (108) and then to the inter-chip connect (148) to other chips in an example system. The scalar unit contains an instruction memory (104), an instruction decode and issue (102), scalar processing unit (107) that contains scalar registers, i.e. 32-bit, a scalar memory (104), and two ALUs (106a,b) for performing two operations per clock cycle. The scalar unit can feed operands and immediate values into the vector operations. Each instruction can be sent from the instruction decode and issue (102) as an instruction bundle that contains the instructions that execute on the vector registers (106). Each instruction bundle is a very long instruction word (VLIW) with each instruction being a number of bits wide, divided into a number of instruction fields.

FIG. 2 illustrates an example architecture 200 including a matrix multiply unit. The matrix multiply unit is a two-dimensional systolic array 206. The array 206 is wired to perform matrix multiplies and typically has a uniform structure throughout the array. A matrix multiply unit is composed of multiply-add sub-unit cells, each of which take an input operand, multiply the operand by a stored weight to obtain a result, and add the result to a partial sum to produce a new partial sum. In some implementations, the sub-unit cells can be grouped into larger multi-cells, i.e. 2 x 2 arrays of multiply-add sub-unit cells or 4 x 4 arrays of multiply-add sub-unit cells. Instead of moving input data from one multiply-add sub-unit cell to the next at a rate of one per clock cycle, the data moves across the systolic array at one multi-cell per clock cycle. As illustrated, the array 206 includes multiple cells grouped as multi-cells 204.

In some implementations, a first dimension 220 of the systolic array 206 corresponds to columns of multi-cells and a second dimension 222 of the systolic array 206 corresponds to rows of multi-cells. The systolic array 206 can have more rows than columns, more columns than rows, or an equal number of columns and rows.

In the illustrated example, value loaders 202 send activation inputs to rows of the array 206 and a weight fetcher interface 208 sends weight inputs to columns of the array 206. In some other implementations, however, activation inputs are transferred to the columns and weight inputs are transferred to the rows of the array 206.

The value loaders 202 can receive the activation inputs from a memory or from a vector processing unit. Each value loader can send multiple corresponding activation inputs to a distinct left-most multi-cell of the array 206. The left-most multi-cell can be a multi-cell along a left-most column of the array 206. For example, value loader 212 can send multiple activation inputs to multi-cell 214. Each activation input received by a multi-cell may be stored in a corresponding activation register of the multi-cell, such as activation registers 315 described in more detail below with reference to FIG. 3. The value loader can also send the multiple activation inputs to an adjacent value loader, such that the multiple activation inputs can be used at another left-most multi-cell of the array 206, i.e. multi-cell 218. This process allows activation inputs to be shifted for use in another particular multi-cell of the array 206.

The weight fetcher interface 208 can receive weight input values from a memory unit, e.g., the vector memory 108 of FIG. 1. The weight fetcher interface 208 can send multiple corresponding weight input values to a distinct top-most multi-cell of the array 206. The top-most multi-cell can be a multi-cell along a top-most row of the array 206. For example, the weight fetcher interface 208, can send a first group of weight input values to multi-cell 214 and a second group of weight inputs to multi-cell 216.

In some implementations, a host interface, e.g., the host interface 150 of FIG. 1, shifts activation inputs throughout the array 206 along one dimension, e.g., to the right, while shifting weight input values throughout the array 206 along another dimension, e.g., to the bottom. For example, over one clock cycle, each activation input of the multiple activation inputs at multi-cell 214 can shift to a corresponding activation register in multi-cell 216, which is to the right of multi-cell 214. Similarly, each weight input value of the multiple weight inputs at multi-cell 216 can shift to a corresponding weight register at multi-cell 218, which is below multi-cell 214.

On each clock cycle, each multi-cell can process the multiple given weight input values and the multiple given activation inputs to generate multiple accumulated outputs. The accumulated outputs can also be passed to an adjacent multi-cell along the same dimension as the given weight input values. An individual multi-cell is described further below with reference to FIG. 3. Each multi-cell includes multiple rows and columns of sub-unit cells. Each row within the multi-cell receives different activation inputs. Each column within the multi-cell receives different weight inputs.

In some implementations, weight sequencers 224 configure whether weight inputs shift to adjacent multi-cells. A weight sequencer 226 can receive a control value from a host, e.g., the host interface 150 of FIG. 1, or an external processor. Each weight sequencer can pass the control value to a corresponding multi-cell in the array 206. In particular, the control value can be stored in a weight control register in the multi-cell. The control value can determine whether weight inputs are shifted along a dimension of the array or loaded. The weight sequencer can also send the control value to an adjacent weight sequencer, which can regulate shifting or loading a corresponding weight input for a corresponding multi-cell.

FIG. 3 illustrates an example architecture of a multi-cell inside a matrix multiply unit. As discussed above, the matrix multiply unit is a two-dimensional systolic array. The array includes multiple multiply-add sub-units that can be grouped into multi-cells. In some implementations, a first dimension of the systolic array corresponds to columns of cells and a second dimension of the systolic array corresponds to rows of cells. The systolic array can have more rows than columns, more columns than rows, or an equal number of columns and rows. This specification describes certain processing for columns or vertically. However, different designs can perform the processing for rows or horizontally.

In the illustrated example, activation data registers 315a, 315b send activation inputs to rows of the array. Weight shift chains 301A and 301B send weight input values to columns of the array, and weight shift chains 302a and 302b send weight input values to rows of the array. A shift chain is a wired pathway along which values can be passed, e.g., from a memory and to each of various registers within the matrix multiply unit. Each weight shift register 305 is designed to shift its weight content value to a weight register 325 so that a matrix multiply unit can send a weight value from memory to a shift register 325 and then to the weight register to be used in a multiplication process of the matrix multiply unit.

The activation data registers 315a, 315b can receive the activation inputs. Each activation data register can send multiple corresponding activation inputs to the multi-cell 300 of the array. Each activation input received by a multi-cell may be stored in a corresponding activation register of the multi-cell, such as the activation data registers 315a, 315b. The activation registers store activation inputs which may be provided by the unified buffer or by an adjacent multi-cell located to the left of the given multi-cell, depending on the position of the multi-cell within the array. For instance, if the multi-cell 300 is located at the left most position within the systolic array of the matrix multiply unit, the activation inputs are provided by the unified buffer. The unified buffer may provide multiple different activation inputs to the multi-cell 300, in which each received activation input then may be stored by a different one of the activation data registers 315. For instance the multi-cell 300 may receive two activation inputs from the unified buffer (e.g. every clock cycle), in which the two activation inputs are stored by the two activation data registers 315, respectively.

Each activation register may be coupled to cells along a first dimension of the array of multi-cells. The connection of the activation registers to the cells is indicated by dotted lines in FIG. 3. For example, activation data register 315a (an activation register) in the multi-cell is coupled to the cells 350a and 350c of the first row. Similarly, activation data register 315b (a second activation register) in the multi-cell is coupled to the cells 350b and 350d of the second row. Each activation register 315 transfers the stored activation input to the cells 350 to which the activation register is coupled. Thus, for a given number of cells extending along a first dimension (e.g., along a given row or along a given column), the activation inputs can be passed to all cells in the multi-cell, and not just a single cell, thereby causing the activation input to spread quickly throughout the array of cells, improving the efficiency of operation of the multi-cell.

The unified buffer can also send the multiple activation inputs to an adjacent activation register so that multiple activation inputs can be used at another multi-cell of the array. This process allows activation inputs to be shifted for use in another particular multi-cell of the array.

Each cell 350 of a multi-cell 300 contains a stored weight value. Before beginning a matrix multiply process, weights are loaded by shifting them into the cells of the systolic array. Dedicated chains and weight shift registers are provided for weight shifting so that new weights can be shifted in concurrently with the execution of previous matrix multiply processing. Weight inputs can be loaded into multi-cells in ways that lower the latency of the overall matrix multiply operational processing.

As discussed above, the weight shift chains 301, 302 can receive weight inputs from a memory unit, e.g., the vector memory 108 of FIG. 1. The shift chains can send multiple corresponding weight inputs to the weight registers 325 associated with the multi-cell 300.

In some implementations, a host interface, e.g., the host interface 150 of FIG. 1, shifts activation inputs throughout the array along one dimension, e.g., to the right, while shifting weight input throughout the array along one or both dimensions, e.g., to the right or to the bottom. For example, over one clock cycle, each activation input of the multiple activation inputs at multi-cell 300 can shift to a corresponding activation register in the next multi-cell in the same row. Similarly, each weight input of the multiple weight inputs at multi-cell 300 can shift to a corresponding weight register at a multi-cell to the right of multi-cell 300 and/or down from multi-cell 300.

A multiplexer 330 selects a weight either from the first shift chain 301 or the second shift chain 302 and forwards the selected input into a single line into the weight shift register 325. Although multiplexers 330 are shown outside of the cell 350 boundary lines, in some implementations the multiplexers 330 exist within the cells 350.

On a clock cycle, each multi-cell can process the multiple given weight inputs and the multiple given activation inputs to generate multiple accumulated outputs. Generally, processing includes a multiplication operation to multiply an activation input with a stored weight. The accumulated outputs can also be passed to an adjacent multi-cell along the same dimension as the given weight inputs. In some implementations, weights and activations are shifted more than more than one multi-cell during a given clock cycle to transition from one convolution calculation to another.

The accumulated outputs can be passed along the same columns as weight inputs, e.g., towards the bottom of the column in the array. In some implementations, a partial sum register 310a, 311A passes a partial sum value into the multi-cell from a previous multi-cell. The array can include partial sum registers 310b, 311B that store the accumulated outputs from each column of multi-cells. For each column of the multi-cell, the products generated by the sub-unit cells in the column are combined with the incoming partial sum from and then sent on as the next partial sum. For certain multi-cells, e.g., the multi-cells in the bottom column of the systolic array, the accumulated outputs may include final accumulated values that can be transferred to a vector computation unit. In some implementations, the final accumulated values are transferred directly from the bottom multi-cells of the array to the vector computation unit while in other implementations, the final accumulated values are first stored in a memory or are processed by a different component before being sent to the vector computation unit.

FIGS. 4A-4B show example architectures 400-450 that each include a matrix computation unit.

The matrix computation unit depicted in FIG. 4A is a two-dimensional systolic array 406. The array 406 includes multiple cells 404. In some implementations, a first dimension

420 of the systolic array 406 corresponds to columns of cells and a second dimension 422 of the systolic array 406 corresponds to rows of cells. The systolic array can have more rows than columns, more columns than rows, or an equal number of columns and rows.

In the illustrated example, value loaders 402 send activation inputs to rows of the array 406 and a weight fetcher interface 408 sends weight inputs to columns of the array 406. In some other implementations, however, activation inputs are transferred to the columns and weight inputs are transferred to the rows of the array 406.

The value loaders 402 can receive the activation inputs from a unified buffer, e.g., the unified buffer 308 of FIG. 3. Each value loader can send a corresponding activation input to a distinct left-most cell of the array 406. For example, value loader 412 can send an activation input to cell 414.

The weight fetcher interface 408 can receive the weight input from a memory unit, e.g., the dynamic memory 310 of FIG. 3. The weight fetcher interface 408 can send a corresponding weight input to a distinct top-most cell of the array 406. For example, the weight fetcher interface 408 can send weight inputs to cells 414 and 416. The weight fetcher interface 408 is further capable of receiving multiple weights from the memory unit, e.g., the dynamic memory 310, and of sending the multiple weights to distinct top-most cells of the array 406 in parallel. For example, the weight fetcher interface 408 may send different weights to the cells 414 and 416 simultaneously.

In some implementations, a host interface, e.g., the host interface 302 of FIG. 3, shifts activation inputs throughout the array 406 along one dimension, e.g., to the right, while shifting weight inputs throughout the array 406 along another dimension, e.g., to the bottom. For example, over one clock cycle, the activation input at cell 414 can shift to an activation register in cell 416, which is to the right of cell 414. Similarly, the weight input at cell 416 can shift to a weight register at cell 418, which is below cell 414.

On each clock cycle, each cell can process a given weight input, a given activation input, and an accumulated output from an adjacent cell to generate an accumulated output. The accumulated output can also be passed to the adjacent cell along the same dimension as the given weight input. Each cell may also process a given weight input and a given activation input to generate an output, without processing an accumulated output from an adjacent cell. The output can be passed to adjacent cells along the same dimensions as the given weight input and output without being accumulated. An individual cell is described further below with reference FIG. 5.

In some implementations, on each clock cycle, each cell multiplies a given weight input and activation input to generate a product value. A cell can then combine the product value with a partial sum value received from another cell to generate an updated partial sum value. The cell can then transmit the partial sum value to another cell in the matrix computation unit.

In some implementations, an identity matrix, i.e., a matrix having ones on the principal diagonal and zeros elsewhere, can be passed to the array 406, thereby passing the inputs provided at the value loaders 402 to the accumulators 410 without modification. This may be used to perform element-wise multiplication of two inputs, where a first output at the accumulators can be represented as output = MatMul(input1, identity), where MatMul is an instruction for the matrix computation unit to perform matrix multiplication, and a second output corresponding to the element-wise multiplication result is represented as output *= MatMul(input2, identity). To perform the *= operation, i.e., the operation output = output * MatMul(input2, identity), the architecture 400 may include a component for performing a += or *= computations. The component for performing the += or *= operations may be positioned before the accumulators 410, i.e., after the last row of cells 404. In some implementations, the vector computation unit 314 of FIG. 3 may include the component for performing the += or *= operations, i.e., where the vector computation unit 314 performs the output *= MatMul(input2,identity) operation to perform element-wise multiplication.

The accumulated output can be passed along the same column as the weight input, e.g., towards the bottom of the column in the array 406. In some implementations, at the bottom of each column, the array 406 can include accumulator units 410 that store and accumulate each accumulated output from each column when performing calculations with layers having more activation inputs than rows. In some implementations, each accumulator unit stores multiple parallel accumulations. The accumulator units 410 can accumulate each accumulated output to generate a final accumulated value. The final accumulated value can be transferred to a vector computation unit, e.g., the vector computation unit of FIG. 6. In some other implementations, the accumulator units 410 passes the accumulated values to the vector computation unit without performing any accumulations when processing layers with layers having fewer activating inputs than rows.

FIG. 4B illustrates an example architecture 450 of a multi-cell inside a matrix computation unit. The matrix computation unit is a two-dimensional systolic array. The array includes multiple combinations of multiplication circuitries and summation circuitries that can be grouped into multi-cells. In some implementations, a first dimension of the systolic array corresponds to columns of cells and a second dimension of the systolic array corresponds to rows of cells. The systolic array can have more rows than columns, more columns than rows, or an equal number of columns and rows. This specification describes certain processing for columns or vertically. However, different designs can perform the processing for rows or horizontally.

In the illustrated example, activation data registers 465a, 465b send activation inputs to rows of the array. Weight shift chains 451A and 451B send weight input values to columns of the array, and weight shift chains 452a and 452b send weight input values to rows of the array. A shift chain is a wired pathway along which values can be passed, e.g., from a memory and to each of various registers within the matrix computation unit. Each weight shift register 455 is designed to shift its weight content value to a weight register 4755 so that a matrix computation unit can send a weight value from memory to a shift register 465 and then to the weight register to be used in a multiplication process of the matrix computation unit.

The activation data registers 465a, 465b can receive the activation inputs from a unified buffer, e.g., the unified buffer 308 of FIG. 3. Each activation data register can send multiple corresponding activation inputs to the multi-cell 450 of the array. Each activation input received by a multi-cell may be stored in a corresponding activation register of the multi-cell, such as the activation data registers 465a, 465b. The activation registers store activation inputs which may be provided by the unified buffer or by an adjacent multi-cell located to the left of the given multi-cell, depending on the position of the multi-cell within the array. For instance, if the multi-cell 450 is located at the left most position within the systolic array of the matrix computation unit, the activation inputs are provided by the unified buffer. The unified buffer may provide multiple different activation inputs to the multi-cell 450, in which each received activation input then may be stored by a different one of the activation data registers 465. For instance, the multi-cell 450 may receive two activation inputs from the unified buffer (e.g. every clock cycle), in which the two activation inputs are stored by the two activation data registers 465, respectively.

Each activation register may be coupled to cells along a first dimension of the array of multi-cells. The connection of the activation registers to the cells is indicated by dotted lines in FIG. 4. For example, activation data register 465a (an activation register) in the multi-cell is coupled to the cells 490a and 490b of the first row. Similarly, activation data register 465b (a second activation register) in the multi-cell is coupled to the cells 49b and 490d of the second row. Each activation register 465 transfers the stored activation input to the cells 350 to which the activation register is coupled. Thus, for a given number of cells extending along a first dimension (e.g., along a given row or along a given column), the activation inputs can be passed to all cells in the multi-cell, and not just a single cell, thereby causing the activation input to spread quickly throughout the array of cells, improving the efficiency of operation of the multi-cell.

The unified buffer can also send the multiple activation inputs to an adjacent activation register so that multiple activation inputs can be used at another multi-cell of the array. This process allows activation inputs to be shifted for use in another particular multi-cell of the array.

Each cell 490 of a multi-cell 450 contains a stored weight value. Before beginning a matrix multiply process, weights are loaded by shifting them into the cells of the systolic array. Dedicated chains and weight shift registers are provided for weight shifting so that new weights can be shifted in concurrently with the execution of previous matrix multiply processing. Weight inputs can be loaded into multi-cells in ways that lower the latency of the overall matrix multiply operational processing.

As discussed above, the weight shift chains 451, 452 can receive weight inputs from a memory unit, e.g., the dynamic memory 310 of FIG. 3. The shift chains can send multiple corresponding weight inputs to the weight registers 325 associated with the multi-cell 300.

In some implementations, a host interface, e.g., the host interface 302 of FIG. 3, shifts activation inputs throughout the array along one dimension, e.g., to the right, while shifting weight input throughout the array along one or both dimensions, e.g., to the right or to the bottom. For example, over one clock cycle, each activation input of the multiple activation inputs at multi-cell 450 can shift to a corresponding activation register in the next multi-cell in the same row. Similarly, each weight input of the multiple weight inputs at multi-cell 450 can shift to a corresponding weight register at a multi-cell to the right of multi-cell 450 and/or down from multi-cell 300.

A multiplexer 480 selects a weight either from the first shift chain 451 or the second shift chain 452 and forwards the selected input into a single line into the weight shift register 475. Although multiplexers 480 are shown outside of the cell 350 boundary lines, in some implementations the multiplexers 480 exist within the cells 490.

On a clock cycle, each multi-cell can process the multiple given weight inputs and the multiple given activation inputs to generate multiple accumulated outputs. Generally, processing includes a multiplication operation to multiply an activation input with a stored weight. The accumulated outputs can also be passed to an adjacent multi-cell along the same dimension as the given weight inputs. In some implementations, weights and activations are shifted more than more than one multi-cell during a given clock cycle to transition from one convolution calculation to another.

The accumulated outputs can be passed along the same columns as weight inputs, e.g., towards the bottom of the column in the array. In some implementations, a partial sum register 460a, 461A passes a partial sum value into the multi-cell from a previous multi-cell. The array can include partial sum registers 460b, 461B that store the accumulated outputs from each column of multi-cells. For each column of the multi-cell, the products generated by the sub-unit cells in the column are combined with the incoming partial sum from and then sent on as the next partial sum. For certain multi-cells, e.g., the multi-cells in the bottom column of the systolic array, the accumulated outputs may include final accumulated values that can be transferred to a vector computation unit. In some implementations, the final accumulated values are transferred directly from the bottom multi-cells of the array to the vector computation unit while in other implementations, the final accumulated values are first stored in a memory or are processed by a different component before being sent to the vector computation unit.

FIG. 5 shows an example architecture 700 of a cell inside a systolic array, e.g., the systolic array 406 of FIG. 4A or the systolic array 450 of FIG. 4B.

The cell can include an activation register 506 that stores an activation input. The activation register can receive the activation input from a left adjacent cell, i.e., an adjacent cell located to the left of the given cell, or from a unified buffer, depending on the position of the cell within the systolic array. The cell can include a weight register 502 that stores a weight input. The weight input can be transferred from a top adjacent cell or from a weight fetcher interface, depending on the position of the cell within the systolic array. The cell can also include a sum in register 504. The sum in register 504 can store an accumulated value from the top adjacent cell. The activation register 506 and the weight register 502 can be registers that are configured to store values of a particular size, such as floating point values of a particular format.

Multiplication circuitry 508 can be used to multiply the weight input from the weight register 502 with the activation input from the activation register 506. The multiplication circuitry 508 can output the product to summation circuitry 510. In some implementations, the input and output values of the multiplication circuitry 508 may be of different sizes and/or formats.

The summation circuitry 510 can sum the product and the accumulated value from the sum in register 504 to generate a new accumulated value. The summation circuitry 510 can then send the new accumulated value to another sum in register located in a bottom adjacent cell. The new accumulated value can be used as an operand for a summation in the bottom adjacent cell. The summation circuitry 510 can also accept a value from the sum in register 504 and send the value from the sum in register 504 to a bottom adjacent cell without summing the value from the sum in register 504 with the product from the multiplication circuitry 508. In some implementations, the input values of the summation circuitry 510 may be of different sizes and/or formats. In some implementations, some input and output values of the summation circuitry 510 may be of different sizes and/or formats.

The cell can also shift the weight input and the activation input to adjacent cells for processing. For example, the weight path register 512 can send the weight input to another weight register in the bottom adjacent cell. The activation register 506 can send the activation input to another activation register in the right adjacent cell. Both the weight input and the activation input can therefore be reused by other cells in the array at a subsequent clock cycle.

In some implementations, the cell also includes a control register. The control register can store a control signal that determines whether the cell should shift either the weight input or the activation input to adjacent cells. In some implementations, shifting the weight input or the activation input takes one or more clock cycles. The control signal can also determine whether the activation input or weight inputs are transferred to the multiplication circuitry 508, or can determine whether the multiplication circuitry 508 operates on the activation and weight inputs. The control signal can also be passed to one or more adjacent cells, e.g., using a wire.

In some implementations, weights are pre-shifted into a weight path register 512. The weight path register 512 can receive the weight input, e.g., from a top adjacent cell, and transfer the weight input to the weight register 502 based on the control signal. The weight register 502 can statically store the weight input such that as activation inputs are transferred to the cell, e.g., through the activation register 506, over multiple clock cycles, the weight input remains within the cell and is not transferred to an adjacent cell. Therefore, the weight input can be applied to multiple activation inputs, e.g., using the multiplication circuitry 508, and respective accumulated values can be transferred to an adjacent cell.

FIG. 6 shows an example architecture 700 of a vector computation unit 602. The vector computation unit 602 can receive a vector of accumulated values from a matrix computation unit, e.g., the matrix computation unit 312 described in reference to FIG. 3 or the accumulators 410 of the matrix computation unit of FIG. 4.

The vector computation unit 602 can process the vector of accumulated values at the activation unit 604. In some implementations, the activation unit includes circuitry that applies a non-linear function to each accumulated value to generate activation values. For example, the non-linear function can be tanh(x), where x is an accumulated value.

Control signals 610 can be transferred, e.g., by the sequencer 306 of FIG. 3, and can regulate how the vector computation unit 602 processes the vector of accumulated values. That is, the control signals 610 can regulate whether the activation values are pooled, where the activation values are stored, e.g., in the unified buffer 308, or can otherwise regulate handling of the activation values. The control signals 610 can also specify the activation or pooling functions, as well as other parameters for processing the activation values or pooling values, e.g., a stride value.

The vector computation unit 602 can send values, e.g., activation values or pooled values to a unified buffer, e.g., the unified buffer 308 of FIG. 3. In some implementations, the pooling circuitry 608 receives the activation values or pooled values and stores the activation values or pooled values in the unified buffer.

FIG. 7 shows an example format 700 for a floating point value. Each of the values processed by a matrix computation unit, e.g., the values stored by registers of cells of a matrix computation unit, may be stored as a floating point value with a particular format.

The format 700 is characterized by a sign bit 701, a group of bits known as significand 702, and another group of bits known as an exponent 703.

The sign bit 701 indicates whether a value stored using the format 700 is positive or negative. The significand 702 includes one or more significant digits of a value stored using the format 700. Therefore, the size, i.e., number of bits, of the significand 702 of the format 700 represents a maximum possible precision of any value stored using the format 700. The exponent 703 represents the power of a scaling base used to convert the stored value into a normalized form. Therefore, the size of the exponent 703 of the format 700 represents a maximum possible dynamic range of any value stored using the format 700.

In some cases, the normalized form that the system uses to interpret floating point numbers contains one or more constant values. For example, the normalized form can always be the form 1.XXXX * 2^XXXX, where the integer portion of the first value is always constant, e.g., equal to 1. In some such cases, the significand 702 may only include the non-constant bits of the normalized value and not include the constant bits. In these cases, the bits of the normalized form that are constant and thus do not appear in the significand 702 are said to be “hidden bits.” A computer system interpreting a binary floating point value having hidden bits will add the hidden bits to the significand 702 in accordance with the normalized form underlying the format 700 of the value.

The manner in which a computer system stores a binary number as a floating point value depends on the normalized form that the system uses to interpret floating point numbers and the size of the significand 702 and the exponent 703 of the particular floating point format 700 used. For example, a floating point format 700 may include a sign bit 701, 4 bits of significand 702, and 3 bits of exponent 702, and a computer system can interpret a binary number having this format 700 by assuming that the number has the normalized form

X.XXX * 2^XXX, where X is a single binary number, i.e., a "0" or a "1". Moreover, the computer system can assume that that the binary value before the decimal point in the first value of the normalized form is a hidden bit that is always one and does not appear in the significand 702. Thus, such a computer system can store and interpret the binary number +11.111 with a floating point value having a sign bit 701 of "0" (because the binary number is positive), an exponent 703 of "001," and a significand 702 of 1111.

If a computer system cannot properly store a number using a format, an attempt to store the number can cause an overflow and lead to unpredictable or undesirable behavior. The example above illustrates that a computer system can store a binary number whose number of digits exceed the maximum precision allowed in the significand 702 of the format 700 adopted by the system without an overflow through rounding the digits. Even though such rounding leads to reduced precision, it will not cause an overflow.

On the other hand, if the dynamic range of a binary number exceeds the maximum range allowed in the exponent 703 of the format 700, the computer system cannot round the dynamic range. For example, the computer system cannot store and interpret the number 111111111.01 because the normalized form of that number has a dynamic range of 1000, and this dynamic range cannot be represented in the range of values allowed in the exponent 703 of the format 700.

To reduce the possibility of an overflow, if an operation in a computer system in any way transforms a first floating point value having a first format to a second floating point value having a second format, it is important that the dynamic range of the second format be greater than or equal to the dynamic range of the first format. This includes the circumstances in which the system is converting the first floating point value to the second floating point value and the circumstances in which the system is applying an operation on the first floating point value to generate the second floating point value. For example, if the computer system multiplies two values having a first format to generate a result value having a second format, it is important that the dynamic range of the second format be greater than or equal to the dynamic range of the first format to reduce the possibility of an overflow. If the two values being multiplied have different formats, it is important that the dynamic range of the second format be greater than or equal to the dynamic range of the format having the greater dynamic range to reduce the possibility of an overflow.

Examples of floating points formats 700 include an IEEE single-precision format, a bfloat format, and an expanded bfloat format.

The IEEE single-precision format is a 32-bit format that includes a sign bit 701, 8 bits of exponent 703, and a 23 bits of significand 702. A bfloat format is a 16-bit format that has a sign bit 701, 8 bits of exponent 703, and 7 bits of significand 702. An expanded bfloat format is a 20-bit format that includes a 20-bit format that includes a sign bit 701, 8 bits of exponent 703, and 11 bits of significand 702.

Importantly, all the three formats noted above have the same exponent 702 size and thus the same dynamic range. However, the single-precision format allows for more precision than the expanded bfloat format, and the expanded bfloat format allows for more precision than the bfloat format. To reduce the possibility of overflow but increase precision, a matrix computation unit can store activation input and weight input values in registers holding values of the bfloat format, hold the product of the input values in a register holding values of the expanded bfloat format, and hold the sum of the product value and a partial sum value in a register holding values of the IEEE single precision format.

FIG. 8 shows an example architecture 800 for multiplication circuitry of a matrix computation cell. The matrix computation cell depicted in FIG. 8 multiplies two input values, e.g., an activation input value 801 and a weight input value 802, to generate a resulting value, e.g., a product value 805.

The architecture 800 includes a multiplier 803 that multiplies the significand and the sign bit of the two input values to generate a significand and a sign bit of the resulting value and an adder 804 that adds the exponent of the two input values to generate an exponent of the resulting value. The combination of the significand and the sign bit and the exponent of the resulting value generates the resulting value.

FIG. 9 is a flow diagram of an example process 900 for performing matrix multiplication using a matrix computation unit. The process 900 can be performed by a matrix computation unit, e.g., the matrix computation unit 400 of FIG. 4A or the matrix computation unit 450 of FIG. 4B, to perform matrix multiplication in accordance with an architecture of the matrix computation unit.

A matrix computation unit can perform the process 900 multiple times in parallel to compute a vector output that is a product of a vector and a matrix, e.g., an input vector including multiple activation input values and a weight matrix including multiple activation weight values.

The matrix computation unit obtains an activation input value and a weight input value that both have a first floating point format (902). Obtaining the activation input and the weight input values is described in greater detail below with reference to FIG. 10.

The matrix computation unit multiplies, using a multiplication circuitry of the matrix computation unit, the activation input value and the weight input value to generate a product value having a second floating point format (904). The second floating point format has a higher precision than the first floating point format and has a dynamic range that is at least as large as the dynamic range of the first floating point format. By storing the result of multiplying the activation input value and the weight input value in a format that has a higher precision than the format of those input values, the matrix computation unit reduces the likelihood of lost precision in storing the result of the multiplication. By storing the result of multiplying the activation input value and the weight input value in a format that has a dynamic range that is at least as large as the dynamic range of the format of those input values, the matrix computation unit also reduces the likelihood of overflow in storing the result of the multiplication.

Multiplying the activation input value and the weight input value is described in greater detail below with reference to FIGS. 9-10.

In some implementations, the first floating point format is a 16 bit format with a sign bit, an 8 bit exponent, and a 7 bit significand that optionally does not include a hidden bit in the normalized form of a corresponding binary number, and the second floating point format is a 20 bit format with a sign bit, an 8 bit exponent, and a 11 bit significand that optionally does not include a hidden bit in the normalized form of a corresponding binary number.

The matrix computation unit obtains a partial sum value in a third floating point format (906). The matrix computation unit can obtain the partial sum value from a cell in the matrix computation unit.

The third floating point format has a higher precision than the first floating point format and has a dynamic range that is at least as large as the dynamic range of the first floating point format. Therefore, the partial sum value has a format that allows for greater precision than the format of the input values and a dynamic range that is as at least as great as the dynamic range of the format of the input values.

In some implementations, the third floating point format has a higher precision than the second floating point format. In other words, the three floating point formats can be ranked in terms of their precision in the following order, starting with the format with the highest precision: the third floating point format, the second floating point format, and the first floating point format. In some implementations, the third floating point format has a dynamic range that is at least as great as the dynamic range of the second floating point format.

In some implementations, the third floating point format is an IEEE standard 754 single precision format or other 32 bit format with a sign bit, an 8 bit exponent, and a 23 bit significand that does not include a hidden bit in the normalized form of a corresponding binary number.

The matrix computation unit combines, using a summation circuitry of the matrix computation unit, the partial sum value and the product value to generate an updated partial sum value having the third format (908). By storing the result of combining the product value and the partial sum value in the same format as the format of the partial sum value, the matrix computation unit reduces the likelihood of overflow or lost precision. This is especially the case in implementations in which the format of the product value, i.e., the second format, has a lower precision than the format of the updated partial sum value, i.e., the third format. In such implementations, the matrix computation unit reduces the likelihood of lost precision by storing the result of the combination in a format that has a higher precision than the format of the product value. Similarly, in implementations in which the third format has a dynamic range that is at least as great as the dynamic range of the second format, the matrix computation unit reduces the likelihood of overflow by storing the result of the combination in a format that has a greater dynamic range than the format of the product value.

In some implementations, the matrix computation unit transmits the updated partial sum to another component of the matrix computation unit, e.g., a cell in the matrix computation unit or a multi-cell structure in the matrix computation unit.

FIG. 10 is a flow diagram of an example process 1000 for performing a multiplication of an activation input value by a weight input value. For convenience, the process 1000 will be described as being performed by a system of one or more computers located in one or more locations. For example, a neural network processing system, e.g., the neural network processing system 100 of FIG. 1, appropriately programmed in accordance with this specification, can perform the process 1000.

A neural network system can perform the process 1000 multiple times in parallel to compute a vector output that includes the higher-precision portion of the product of a vector and a matrix, e.g., an input vector including multiple activation input values and a weight matrix including multiple activation weight values, and a vector outputs that includes the lower-precision portion of the product of the vector and the matrix.

The system obtains a raw activation value and a raw weight value (1002). The system may obtain the raw values from a neural network implementation engine of the system, e.g., the neural network implementation engine 150 of FIG. 1. The raw values may be in any format, such as an IEEE single-precision floating point format.

The system converts the raw activation value to a first format to generate an activation input value (1004) and converts the raw weight value to the first format to generate a weight input value (1006). The system can store the number represented by the raw activation value as a new value with a new format.

The system determines if it has received a request to multiply the raw activation value and the activation input value with enhanced precision (1008). The system may receive this enhanced precision request from an end user of the system and/or by a neural network implementation engine of the system, e.g., the neural network implementation engine 150 of FIG. 1. The request indicates that the system must store the result of multiplying the raw activation value and the activation input value with reduced loss of precision.

In some implementations, the system receives an enhanced precision request through a neural network implementation engine of the system, e.g., the neural network implementation engine 150 of FIG. 1. If the system determines that it has not received an enhanced precision request, the system multiplies, using a matrix computation unit of a hardware circuitry on which the system is implemented, the activation input value and the raw activation value as individual values to generate a product value having a second format (1010).

Otherwise, if the system determines that it has received an enhanced precision request, the system generates an activation enhanced precision value that is the difference between the raw activation value and the activation input value (1012) and generates a weight enhanced precision value that is the difference between the raw weight value and the weight input value (1014). The system generates the difference between the activation input value and the raw input value by subtracting the activation input value from the raw activation value and generates the difference between the weight input value and the raw weight value by subtracting the weight input value from the raw weight value. The system can do the subtraction of two values using appropriate circuitry outside the matrix computation unit, such as using summation circuitry outside the matrix computation unit by adding a first value to a negation of a second value. The activation enhanced precision value and the weight enhanced precision value are both values in the first floating point format.

The system performs, using the matrix computation unit, a set of multiplications between (1016) the activation input value, the weight input value, the activation enhanced precision value, and the weight enhanced precision value. To perform a multiplication between two values using the matrix computation unit, the system provides the two values to the matrix computation unit to cause the matrix computation unit to perform a multiplication of the two values.

In some implementations, the set of multiplications include: multiplying the activation input value by the weight input value, multiplying the activation input value by the weight enhanced precision value, multiplying the weight input value by the activation enhanced precision value, and multiplying the activation enhanced precision value by the weight enhanced precision value.

In some implementations, the set of multiplications only include multiplying the activation input value by the weight input value and multiplying the activation enhanced precision value by the weight enhanced precision value. This technique can be used to reduce the number of required multiplications when the system determines that at least some of the individual input values and enhanced precision values have a required level of exactness.

The enhanced precision values indicate at least some of the lower-precision portion of the raw values that were lost in rounding when storing the raw values in the input values that have the first format. By using those enhanced precision values in a multiplication, the system can perform multiplications that involve the higher precision portion of the raw values and thus produce a multiplication result that has a greater precision. Multiplying with enhanced precision is described in greater detail below with reference to FIG. 11.

The system then combines the products of the four multiplications to generate (1018) a first value in the first format that that includes the lower precision portion of the result of multiplying the raw values and a second value in the first format that includes the higher portion of the result of multiplying the raw values. In some implementations, the system performs the summation using an external summation circuitry that is external to the matrix computation unit.

Embodiments of the subject matter and the functional operations described in this specification can be implemented in digital electronic circuitry, in tangibly-embodied computer software or firmware, in computer hardware, including the structures disclosed in this specification and their structural equivalents, or in combinations of one or more of them. Embodiments of the subject matter described in this specification can be implemented as one or more computer programs, i.e., one or more modules of computer program instructions encoded on a tangible non transitory program carrier for execution by, or to control the operation of, data processing apparatus. Alternatively or in addition, the program instructions can be encoded on an artificially generated propagated signal, e.g., a machine-generated electrical, optical, or electromagnetic signal, that is generated to encode information for transmission to suitable receiver apparatus for execution by a data processing apparatus. The computer storage medium can be a machine-readable storage device, a machine-readable storage substrate, a random or serial access memory device, or a combination of one or more of them.

The term "data processing apparatus" encompasses all kinds of apparatus, devices, and machines for processing data, including by way of example a programmable processor, a computer, or multiple processors or computers. The apparatus can include special purpose logic circuitry, e.g., an FPGA (field programmable gate array) or an ASIC (application specific integrated circuit). The apparatus can also include, in addition to hardware, code that creates an execution environment for the computer program in question, e.g., code that constitutes processor firmware, a protocol stack, a database management system, an operating system, or a combination of one or more of them.

A computer program (which may also be referred to or described as a program, software, a software application, a module, a software module, a script, or code) can be written in any form of programming language, including compiled or interpreted languages, or declarative or procedural languages, and it can be deployed in any form, including as a standalone program or as a module, component, subroutine, or other unit suitable for use in a computing environment. A computer program may, but need not, correspond to a file in a file system. A program can be stored in a portion of a file that holds other programs or data, e.g., one or more scripts stored in a markup language document, in a single file dedicated to the program in question, or in multiple coordinated files, e.g., files that store one or more modules, sub programs, or portions of code. A computer program can be deployed to be executed on one computer or on multiple computers that are located at one site or distributed across multiple sites and interconnected by a communication network.

The processes and logic flows described in this specification can be performed by one or more programmable computers executing one or more computer programs to perform functions by operating on input data and generating output. The processes and logic flows can also be performed by, and apparatus can also be implemented as, special purpose logic circuitry, e.g., an FPGA (field programmable gate array) or an ASIC (application specific integrated circuit).

Computers suitable for the execution of a computer program include, by way of example, can be based on general or special purpose microprocessors or both, or any other kind of central processing unit. Generally, a central processing unit will receive instructions and data from a read only memory or a random access memory or both. The essential elements of a computer are a central processing unit for performing or executing instructions and one or more memory devices for storing instructions and data. Generally, a computer will also include, or be operatively coupled to receive data from or transfer data to, or both, one or more mass storage devices for storing data, e.g., magnetic, magneto optical disks, or optical disks. However, a computer need not have such devices. Moreover, a computer can be embedded in another device, e.g., a mobile telephone, a personal digital assistant (PDA), a mobile audio or video player, a game console, a Global Positioning System (GPS) receiver, or a portable storage device, e.g., a universal serial bus (USB) flash drive, to name just a few.

Computer readable media suitable for storing computer program instructions and data include all forms of nonvolatile memory, media and memory devices, including by way of example semiconductor memory devices, e.g., EPROM, EEPROM, and flash memory devices; magnetic disks, e.g., internal hard disks or removable disks; magneto optical disks; and CD ROM and DVD-ROM disks. The processor and the memory can be supplemented by, or incorporated in, special purpose logic circuitry.

To send for interaction with a user, embodiments of the subject matter described in this specification can be implemented on a computer having a display device, e.g., a CRT (cathode ray tube) or LCD (liquid crystal display) monitor, for displaying information to the user and a keyboard and a pointing device, e.g., a mouse or a trackball, by which the user can send input to the computer. Other kinds of devices can be used to send for interaction with a user as well; for example, feedback provided to the user can be any form of sensory feedback, e.g., visual feedback, auditory feedback, or tactile feedback; and input from the user can be received in any form, including acoustic, speech, or tactile input. In addition, a computer can interact with a user by sending documents to and receiving documents from a device that is used by the user; for example, by sending web pages to a web browser on a user's client device in response to requests received from the web browser.

Embodiments of the subject matter described in this specification can be implemented in a computing system that includes a back end component, e.g., as a data server, or that includes a middleware component, e.g., an application server, or that includes a front end component, e.g., a client computer having a graphical user interface or a Web browser through which a user can interact with an implementation of the subject matter described in this specification, or any combination of one or more such back end, middleware, or front end components. The components of the system can be interconnected by any form or medium of digital data communication, e.g., a communication network. Examples of communication networks include a local area network ("LAN") and a wide area network ("WAN"), e.g., the Internet.

The computing system can include clients and servers. A client and server are generally remote from each other and typically interact through a communication network.

The relationship of client and server arises by virtue of computer programs running on the respective computers and having a client-server relationship to each other.

While this specification contains many specific implementation details, these should not be construed as limitations on the scope of any invention or of what may be claimed, but rather as descriptions of features that may be specific to particular embodiments of particular inventions. Certain features that are described in this specification in the context of separate embodiments can also be implemented in combination in a single embodiment. Conversely, various features that are described in the context of a single embodiment can also be implemented in multiple embodiments separately or in any suitable subcombination. Moreover, although features may be described above as acting in certain combinations and even initially claimed as such, one or more features from a claimed combination can in some cases be excised from the combination, and the claimed combination may be directed to a subcombination or variation of a subcombination.

Similarly, while operations are depicted in the drawings in a particular order, this should not be understood as requiring that such operations be performed in the particular order shown or in sequential order, or that all illustrated operations be performed, to achieve desirable results. In certain circumstances, multitasking and parallel processing may be advantageous. Moreover, the separation of various system modules and components in the embodiments described above should not be understood as requiring such separation in all embodiments, and it should be understood that the described program components and systems can generally be integrated together in a single software product or packaged into multiple software products.

Particular embodiments of the subject matter have been described. Other embodiments are within the scope of the following claims. For example, the actions recited in the claims can be performed in a different order and still achieve desirable results. As one example, the processes depicted in the accompanying figures do not necessarily require the particular order shown, or sequential order, to achieve desirable results. In certain implementations, multitasking and parallel processing may be advantageous.

What is claimed is:

CLAIMS

1. A method of performing a matrix multiplication using a hardware circuit, the method comprising:

obtaining, by a matrix computation unit of the hardware circuit, an input activation value and a weight input value, the input activation value and the weight input value each having a first floating point format;

multiplying, by a multiplication circuitry of the matrix computation unit, the input activation value and the weight input value to generate a product value, the product value having a second floating point format that has a higher precision than the first floating point format and has a dynamic range that is at least as large as the dynamic range of the first floating point format;

obtaining, by the matrix computation unit, a partial sum value in a third floating point format that has a higher precision than the first floating point format and has a dynamic range that is at least as large as the dynamic range of the first floating point format; and

combining, by a summation circuitry of the hardware circuit, at least the partial sum value and the product value to generate an updated partial sum value that has the third floating point format.

2. The method of claim 1, wherein the precision of a floating point format is determined based on a count of available bits for a significand in the floating point format and the dynamic range of a floating point format is determined based on a count of available bits for an exponent in the floating point format.

3. The method of any one claims 1-2, wherein the second floating point format has the same dynamic range as the first floating point format and the third floating point format has the same dynamic range as the first floating point format.

4. The method of any one of claims 1-3, wherein the third floating point format has a higher precision than the second floating point format.

5. The method of any one of claims 1-4, wherein:

the hardware circuit is configured to perform computations for a neural network having a plurality of layers, and

the input activation value and the weight input value are associated with a layer of the plurality of layers.

6. The method of any one of claims 1-5, further comprising:

obtaining a raw activation value and a raw weight value for the first matrix computation cell having the third floating point format;

converting the raw activation value into the first floating point format to generate the input activation value; and

converting the raw weight value into the first floating point format to generate the weight input value.

7. The method of claim 6, further comprising:

receiving a request to process the raw activation value with enhanced precision;

generating an activation enhanced precision value for the input value, the activation enhanced precision value being a measure of difference between the activation input value and the raw activation value; and

generating a weight enhanced precision value for the weight input value, the weight enhanced precision value being a measure of difference between the weight input value and the raw weight value; and

wherein, multiplying the activation input value and the weight input value to generate the product value comprises:

multiplying, by the multiplication circuitry of the matrix computation unit,

the input value by the weight input value,

the input value by the weight enhanced precision value,

the weight input value by the activation enhanced precision value, and

the activation enhanced precision value by the weight enhanced precision value, and

combining products of the multiplications to generate the product value.

8. A hardware circuit comprising:

a matrix computation unit configured to perform a first group of operations comprising:

obtaining an activation input value and a weight input value, the activation input value and the weight input value both having a first floating point format;

storing the weight input value in a weight register, the weight register being configured to store values having the first floating point format;

multiplying, a using multiplication circuitry of the hardware circuit, the activation input value and the weight input value to generate a product value, the product value having a second floating point format that has a higher precision than the first floating point format and has a dynamic range that is at least as large as the dynamic range of the first floating point format;

obtaining a partial sum value in a third floating point format that has a higher precision than the first floating point format and has a dynamic range that is at least as large as the dynamic range of the first floating point format;

storing the partial sum value in a sum in register, the sum in register being configured to store values having the third floating point format; and

combining, using a summation circuitry of the matrix computation unit, the partial sum value and the product value to generate an updated partial sum value that has the third floating point format.

9. The hardware circuit of claim 8, the first group of operations further comprising:

storing the activation input value in an activation register, the activation register being configured to store values having the first floating point format.

10. The hardware circuit of any one of claims 8-9, the first group of operations further comprising:

storing the weight input value in a weight register, the weight register being configured to store values having the first floating point format.

11. The hardware circuit of any one of claim 8-10, the first group of operations further comprising:

storing the partial sum value in a sum in register, the sum in register being configured to store values having the third floating point format.

12. The hardware circuit of any one of claim 8-11, further comprising an external summation circuitry outside the matrix computation unit, and wherein the first group of operations further comprises:

receiving a request to process the raw activation value with enhanced precision;

generating an activation enhanced precision value for the input value, the activation enhanced precision value being a measure of difference between the activation input value and the raw activation value; and

generating a weight enhanced precision value for the weight input value, the weight enhanced precision value being a measure of difference between the weight input value and the raw weight value; and

wherein, multiplying the activation input value and the weight input value to generate the product value comprises:

multiplying, by the multiplication circuitry of the matrix computation unit,

the input value by the weight input value,

the input value by the weight enhanced precision value,

the weight input value by the activation enhanced precision value, and

the activation enhanced precision value by the weight enhanced precision value, and

wherein the external summation circuitry is configured to perform a second group of operations comprising:

combining products of the multiplications to generate the product value.

13. A system comprising one or more computers and one or more storage devices storing instructions that are operable, when executed by the one or more computers, to cause the one or more computers to perform the operations of the respective method of any one of claims 1-7.

14. A computer storage medium encoded with instructions that, when executed by one or more computers, cause the one or more computers to perform the operations of the respective method of any one of claims 1-7.

15. A system comprising one or more computers and one or more storage devices storing instructions that are operable, when executed by the one or more computers, to cause the one or more computers to perform the operations of the respective hardware circuit of any one of claims 8-12.

16. A computer storage medium encoded with instructions that, when executed by one or more computers, cause the one or more computers to perform the operations of the respective hardware circuit of any one of claims 8-12.

51055083.doc

Application Data Sheet 37 CFR 1.76	Attorney Docket Number	16113-8252P01
	Application Number	
Title of Invention	PERFORMING MATRIX MULTIPLICATION IN HARDWARE	

The application data sheet is part of the provisional or nonprovisional application for which it is being submitted. The following form contains the bibliographic data arranged in a format specified by the United States Patent and Trademark Office as outlined in 37 CFR 1.76.
This document may be completed electronically and submitted to the Office in electronic format using the Electronic Filing System (EFS) or the document may be printed and included in a paper filed application.

Secrecy Order 37 CFR 5.2:

☐ Portions or all of the application associated with this Application Data Sheet may fall under a Secrecy Order pursuant to 37 CFR 5.2 (Paper filers only. Applications that fall under Secrecy Order may not be filed electronically.)

Inventor Information:

Inventor 1 Remove
Legal Name

Prefix	**Given Name**	**Middle Name**	**Family Name**	**Suffix**
	Andrew	Everett	Phelps	

Residence Information (Select One) [✓ US Residency] Non US Residency Active US Military Service

City	Middleton	**State/Province**	WI	**Country of Residence** i	US

Mailing Address of Inventor:

Address 1	1600 Amphitheatre Parkway		
Address 2			
City	Mountain View	**State/Province**	CA
Postal Code	94043	**Country** i	US

Inventor 2 Remove
Legal Name

Prefix	**Given Name**	**Middle Name**	**Family Name**	**Suffix**
	Norman	Paul	Jouppi	

Residence Information (Select One) [✓ US Residency] Non US Residency Active US Military Service

City	Palo Alto	**State/Province**	CA	**Country of Residence** i	US

Mailing Address of Inventor:

Address 1	1600 Amphitheatre Parkway		
Address 2			
City	Mountain View	**State/Province**	CA
Postal Code	94043	**Country** i	US

All Inventors Must Be Listed - Additional Inventor Information blocks may be generated within this form by selecting the **Add** button. Add

Correspondence Information:

Enter either Customer Number or complete the Correspondence Information section below. For further information see 37 CFR 1.33(a).

Application Data Sheet 37 CFR 1.76	Attorney Docket Number	16113-8252P01
	Application Number	
Title of Invention	PERFORMING MATRIX MULTIPLICATION IN HARDWARE	

☐ **An Address is being provided for the correspondence Information of this application.**

Customer Number	26192	
Email Address		Add Email / Remove Email

Application Information:

Title of the Invention	PERFORMING MATRIX MULTIPLICATION IN HARDWARE		
Attorney Docket Number	16113-8252P01	**Small Entity Status Claimed**	☐
Application Type	Provisional		
Subject Matter	Utility		
Total Number of Drawing Sheets (if any)	13	**Suggested Figure for Publication (if any)**	

Filing By Reference:

Only complete this section when filing an application by reference under 35 U.S.C. 111(c) and 37 CFR 1.57(a). Do not complete this section if application papers including a specification and any drawings are being filed. Any domestic benefit or foreign priority information must be provided in the appropriate section(s) below (i.e., "Domestic Benefit/National Stage Information" and "Foreign Priority Information").

For the purposes of a filing date under 37 CFR 1.53(b), the description and any drawings of the present application are replaced by this reference to the previously filed application, subject to conditions and requirements of 37 CFR 1.57(a).

Application number of the previously filed application	Filing date (YYYY-MM-DD)	Intellectual Property Authority or Country

Publication Information:

☐ Request Early Publication (Fee required at time of Request 37 CFR 1.219)

☐ **Request Not to Publish.** I hereby request that the attached application not be published under 35 U.S.C. 122(b) and certify that the invention disclosed in the attached application **has not and will not** be the subject of an application filed in another country, or under a multilateral international agreement, that requires publication at eighteen months after filing.

Representative Information:

Representative information should be provided for all practitioners having a power of attorney in the application. Providing this information in the Application Data Sheet does not constitute a power of attorney in the application (see 37 CFR 1.32).
Either enter Customer Number or complete the Representative Name section below. If both sections are completed the customer Number will be used for the Representative Information during processing.

Please Select One:	● Customer Number	US Patent Practitioner	◯ Limited Recognition (37 CFR 11.9)
Customer Number	26192		

Application Data Sheet 37 CFR 1.76	Attorney Docket Number	16113-8252P01
	Application Number	
Title of Invention	PERFORMING MATRIX MULTIPLICATION IN HARDWARE	

Domestic Benefit/National Stage Information:

This section allows for the applicant to either claim benefit under 35 U.S.C. 119(e), 120, 121, 365(c), or 386(c) or indicate National Stage entry from a PCT application. Providing benefit claim information in the Application Data Sheet constitutes the specific reference required by 35 U.S.C. 119(e) or 120, and 37 CFR 1.78.
When referring to the current application, please leave the “Application Number” field blank.

Prior Application Status			Remove
Application Number	Continuity Type	Prior Application Number	Filing or 371(c) Date (YYYY-MM-DD)

Additional Domestic Benefit/National Stage Data may be generated within this form by selecting the **Add** button. Add

Foreign Priority Information:

This section allows for the applicant to claim priority to a foreign application. Providing this information in the application data sheet constitutes the claim for priority as required by 35 U.S.C. 119(b) and 37 CFR 1.55. When priority is claimed to a foreign application that is eligible for retrieval under the priority document exchange program (PDX)[i] the information will be used by the Office to automatically attempt retrieval pursuant to 37 CFR 1.55(i)(1) and (2). Under the PDX program, applicant bears the ultimate responsibility for ensuring that a copy of the foreign application is received by the Office from the participating foreign intellectual property office, or a certified copy of the foreign priority application is filed, within the time period specified in 37 CFR 1.55(g)(1).

Remove

Application Number	Country[i]	Filing Date (YYYY-MM-DD)	Access Code[i] (if applicable)

Additional Foreign Priority Data may be generated within this form by selecting the **Add** button. Add

Statement under 37 CFR 1.55 or 1.78 for AIA (First Inventor to File) Transition Applications

☐ This application (1) claims priority to or the benefit of an application filed before March 16, 2013 and (2) also contains, or contained at any time, a claim to a claimed invention that has an effective filing date on or after March 16, 2013.
NOTE: By providing this statement under 37 CFR 1.55 or 1.78, this application, with a filing date on or after March 16, 2013, will be examined under the first inventor to file provisions of the AIA.

Application Data Sheet 37 CFR 1.76	Attorney Docket Number	16113-8252P01
	Application Number	
Title of Invention	PERFORMING MATRIX MULTIPLICATION IN HARDWARE	

Authorization or Opt-Out of Authorization to Permit Access:

When this Application Data Sheet is properly signed and filed with the application, applicant has provided written authority to permit a participating foreign intellectual property (IP) office access to the instant application-as-filed (see paragraph A in subsection 1 below) and the European Patent Office (EPO) access to any search results from the instant application (see paragraph B in subsection 1 below).

Should applicant choose not to provide an authorization identified in subsection 1 below, applicant **must opt-out** of the authorization by checking the corresponding box A or B or both in subsection 2 below.

NOTE: This section of the Application Data Sheet is **ONLY** reviewed and processed with the **INITIAL** filing of an application. After the initial filing of an application, an Application Data Sheet cannot be used to provide or rescind authorization for access by a foreign IP office(s). Instead, Form PTO/SB/39 or PTO/SB/69 must be used as appropriate.

1. Authorization to Permit Access by a Foreign Intellectual Property Office(s)

A. Priority Document Exchange (PDX) - Unless box A in subsection 2 (opt-out of authorization) is checked, the undersigned hereby **grants the USPTO authority** to provide the European Patent Office (EPO), the Japan Patent Office (JPO), the Korean Intellectual Property Office (KIPO), the State Intellectual Property Office of the People's Republic of China (SIPO), the World Intellectual Property Organization (WIPO), and any other foreign intellectual property office participating with the USPTO in a bilateral or multilateral priority document exchange agreement in which a foreign application claiming priority to the instant patent application is filed, access to: (1) the instant patent application-as-filed and its related bibliographic data, (2) any foreign or domestic application to which priority or benefit is claimed by the instant application and its related bibliographic data, and (3) the date of filing of this Authorization. See 37 CFR 1.14(h)(1).

B. Search Results from U.S. Application to EPO - Unless box B in subsection 2 (opt-out of authorization) is checked, the undersigned hereby **grants the USPTO authority** to provide the EPO access to the bibliographic data and search results from the instant patent application when a European patent application claiming priority to the instant patent application is filed. See 37 CFR 1.14(h)(2).

The applicant is reminded that the EPO's Rule 141(1) EPC (European Patent Convention) requires applicants to submit a copy of search results from the instant application without delay in a European patent application that claims priority to the instant application.

2. Opt-Out of Authorizations to Permit Access by a Foreign Intellectual Property Office(s)

☐ A. Applicant **DOES NOT** authorize the USPTO to permit a participating foreign IP office access to the instant application-as-filed. If this box is checked, the USPTO will not be providing a participating foreign IP office with any documents and information identified in subsection 1A above.

☐ B. Applicant **DOES NOT** authorize the USPTO to transmit to the EPO any search results from the instant patent application. If this box is checked, the USPTO will not be providing the EPO with search results from the instant application.

NOTE: Once the application has published or is otherwise publicly available, the USPTO may provide access to the application in accordance with 37 CFR 1.14.

Application Data Sheet 37 CFR 1.76	Attorney Docket Number	16113-8252P01
	Application Number	
Title of Invention	PERFORMING MATRIX MULTIPLICATION IN HARDWARE	

Applicant Information:

Providing assignment information in this section does not substitute for compliance with any requirement of part 3 of Title 37 of CFR to have an assignment recorded by the Office.

Applicant 1 Remove

If the applicant is the inventor (or the remaining joint inventor or inventors under 37 CFR 1.45), this section should not be completed. The information to be provided in this section is the name and address of the legal representative who is the applicant under 37 CFR 1.43; or the name and address of the assignee, person to whom the inventor is under an obligation to assign the invention, or person who otherwise shows sufficient proprietary interest in the matter who is the applicant under 37 CFR 1.46. If the applicant is an applicant under 37 CFR 1.46 (assignee, person to whom the inventor is obligated to assign, or person who otherwise shows sufficient proprietary interest) together with one or more joint inventors, then the joint inventor or inventors who are also the applicant should be identified in this section. Clear

Assignee	Legal Representative under 35 U.S.C. 117	Joint Inventor
● Person to whom the inventor is obligated to assign.	Person who shows sufficient proprietary interest	

If applicant is the legal representative, indicate the authority to file the patent application, the inventor is:

Name of the Deceased or Legally Incapacitated Inventor:	
If the Applicant is an Organization check here.	☒
Organization Name	Google Inc.

Mailing Address Information For Applicant:

Address 1	1600 Amphitheatre Parkway		
Address 2			
City	Mountain View	**State/Province**	CA
Country	US	Postal Code	94043
Phone Number	650-253-0000	Fax Number	650-618-1499
Email Address			

Additional Applicant Data may be generated within this form by selecting the Add button. Add

Assignee Information including Non-Applicant Assignee Information:

Providing assignment information in this section does not substitute for compliance with any requirement of part 3 of Title 37 of CFR to have an assignment recorded by the Office.

Application Data Sheet 37 CFR 1.76	Attorney Docket Number	16113-8252P01
	Application Number	
Title of Invention	PERFORMING MATRIX MULTIPLICATION IN HARDWARE	

Assignee 1

Complete this section if assignee information, including non-applicant assignee information, is desired to be included on the patent application publication. An assignee-applicant identified in the "Applicant Information" section will appear on the patent application publication as an applicant. For an assignee-applicant, complete this section only if identification as an assignee is also desired on the patent application publication.

Remove

If the Assignee or Non-Applicant Assignee is an Organization check here. ☒

Organization Name	Google Inc.

Mailing Address Information For Assignee including Non-Applicant Assignee:

Address 1	1600 Amphitheatre Parkway		
Address 2			
City	Mountain View	**State/Province**	CA
Country i	US	Postal Code	94043
Phone Number	650-253-0000	Fax Number	650-618-1499
Email Address			

Additional Assignee or Non-Applicant Assignee Data may be generated within this form by selecting the Add button. Add

Signature:

Remove

NOTE: This Application Data Sheet must be signed in accordance with 37 CFR 1.33(b). **However, if this Application Data Sheet is submitted with the <u>INITIAL</u> filing of the application <u>and</u> either box A or B is <u>not</u> checked in subsection 2 of the "Authorization or Opt-Out of Authorization to Permit Access" section, then this form must also be signed in accordance with 37 CFR 1.14(c).**

This Application Data Sheet **<u>must</u>** be signed by a patent practitioner if one or more of the applicants is a **juristic entity** (e.g., corporation or association). If the applicant is two or more joint inventors, this form must be signed by a patent practitioner, **<u>all</u>** joint inventors who are the applicant, or one or more joint inventor-applicants who have been given power of attorney (e.g., see USPTO Form PTO/AIA/81) on behalf of **<u>all</u>** joint inventor-applicants.

See 37 CFR 1.4(d) for the manner of making signatures and certifications.

Signature	/Stephanie D. Grosvenor/			Date (YYYY-MM-DD)	2017-05-17
First Name	Stephanie	Last Name	Grosvenor	Registration Number	67592

Additional Signature may be generated within this form by selecting the Add button. Add

Application Data Sheet 37 CFR 1.76	Attorney Docket Number	16113-8252P01
	Application Number	
Title of Invention	PERFORMING MATRIX MULTIPLICATION IN HARDWARE	

Privacy Act Statement

The Privacy Act of 1974 (P.L. 93-579) requires that you be given certain information in connection with your submission of the attached form related to a patent application or patent. Accordingly, pursuant to the requirements of the Act, please be advised that: (1) the general authority for the collection of this information is 35 U.S.C. 2(b)(2); (2) furnishing of the information solicited is voluntary; and (3) the principal purpose for which the information is used by the U.S. Patent and Trademark Office is to process and/or examine your submission related to a patent application or patent. If you do not furnish the requested information, the U.S. Patent and Trademark Office may not be able to process and/or examine your submission, which may result in termination of proceedings or abandonment of the application or expiration of the patent.

The information provided by you in this form will be subject to the following routine uses:

1 The information on this form will be treated confidentially to the extent allowed under the Freedom of Information Act (5 U.S.C. 552) and the Privacy Act (5 U.S.C. 552a). Records from this system of records may be disclosed to the Department of Justice to determine whether the Freedom of Information Act requires disclosure of these records.

2. A record from this system of records may be disclosed, as a routine use, in the course of presenting evidence to a court, magistrate, or administrative tribunal, including disclosures to opposing counsel in the course of settlement negotiations.

3 A record in this system of records may be disclosed, as a routine use, to a Member of Congress submitting a request involving an individual, to whom the record pertains, when the individual has requested assistance from the Member with respect to the subject matter of the record.

4. A record in this system of records may be disclosed, as a routine use, to a contractor of the Agency having need for the information in order to perform a contract. Recipients of information shall be required to comply with the requirements of the Privacy Act of 1974, as amended, pursuant to 5 U.S.C. 552a(m).

5. A record related to an International Application filed under the Patent Cooperation Treaty in this system of records may be disclosed, as a routine use, to the International Bureau of the World Intellectual Property Organization, pursuant to the Patent CooperationTreaty.

6. A record in this system of records may be disclosed, as a routine use, to another federal agency for purposes of National Security review (35 U.S.C. 181) and for review pursuant to the Atomic Energy Act (42 U.S.C. 218(c)).

7. A record from this system of records may be disclosed, as a routine use, to the Administrator, General Services, or his/her designee, during an inspection of records conducted by GSA as part of that agency's responsibility to recommend improvements in records management practices and programs, under authority of 44 U.S.C. 2904 and 2906. Such disclosure shall be made in accordance with the GSA regulations governing inspection of records for this purpose, and any other relevant (i.e., GSA or Commerce) directive. Such disclosure shall not be used to make determinations about individuals.

8. A record from this system of records may be disclosed, as a routine use, to the public after either publication of the application pursuant to 35 U.S.C. 122(b) or issuance of a patent pursuant to 35 U.S.C. 151. Further, a record may be disclosed, subject to the limitations of 37 CFR 1.14, as a routine use, to the public if the record was filed in an application which became abandoned or in which the proceedings were terminated and which application is referenced by either a published application, an application open to public inspections or an issued patent.

9. A record from this system of records may be disclosed, as a routine use, to a Federal, State, or local law enforcement agency, if the USPTO becomes aware of a violation or potential violation of law or regulation.

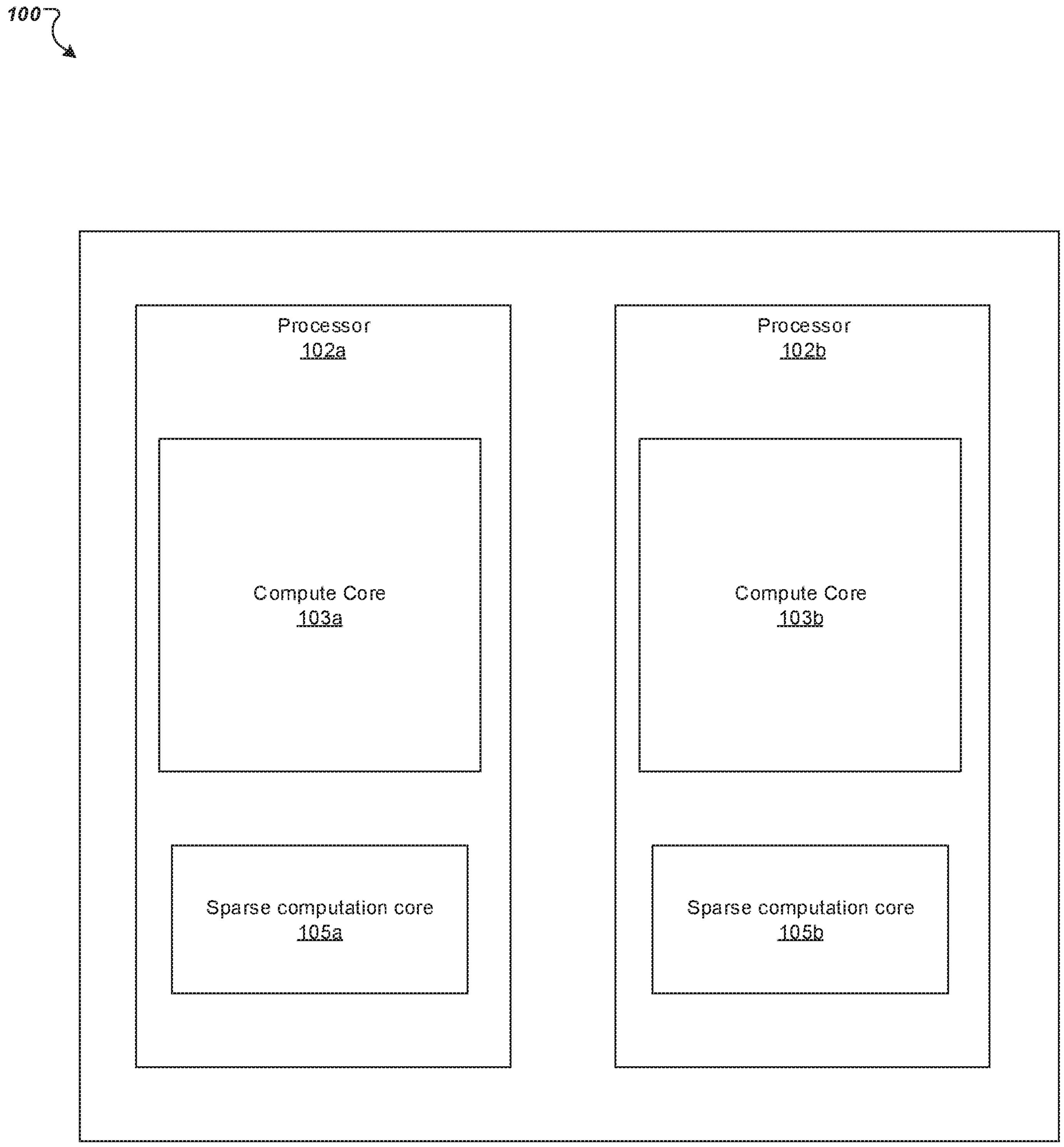
100
Processor
102a
Compute Core
103a
Sparse computation core
105a
Processor
102b
Compute Core
103b
Sparse computation core
105b

FIG. 1A

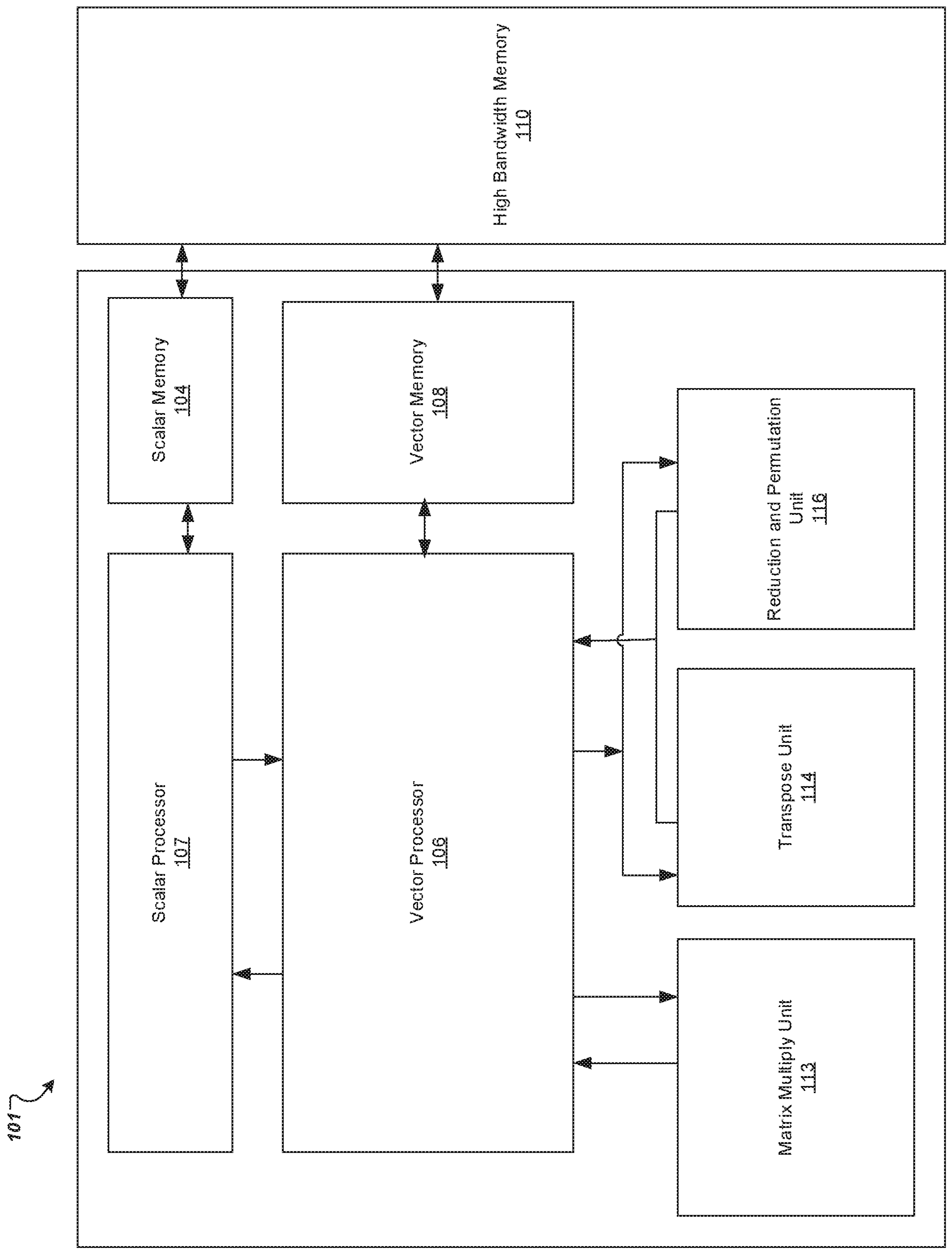
101
High Bandwidth Memory 110
Scalar Memory 104
Vector Memory 108
Scalar Processor 107
Vector Processor 106
Reduction and Permutation Unit 116
Transpose Unit 114
Matrix Multiply Unit 113

FIG. 1B

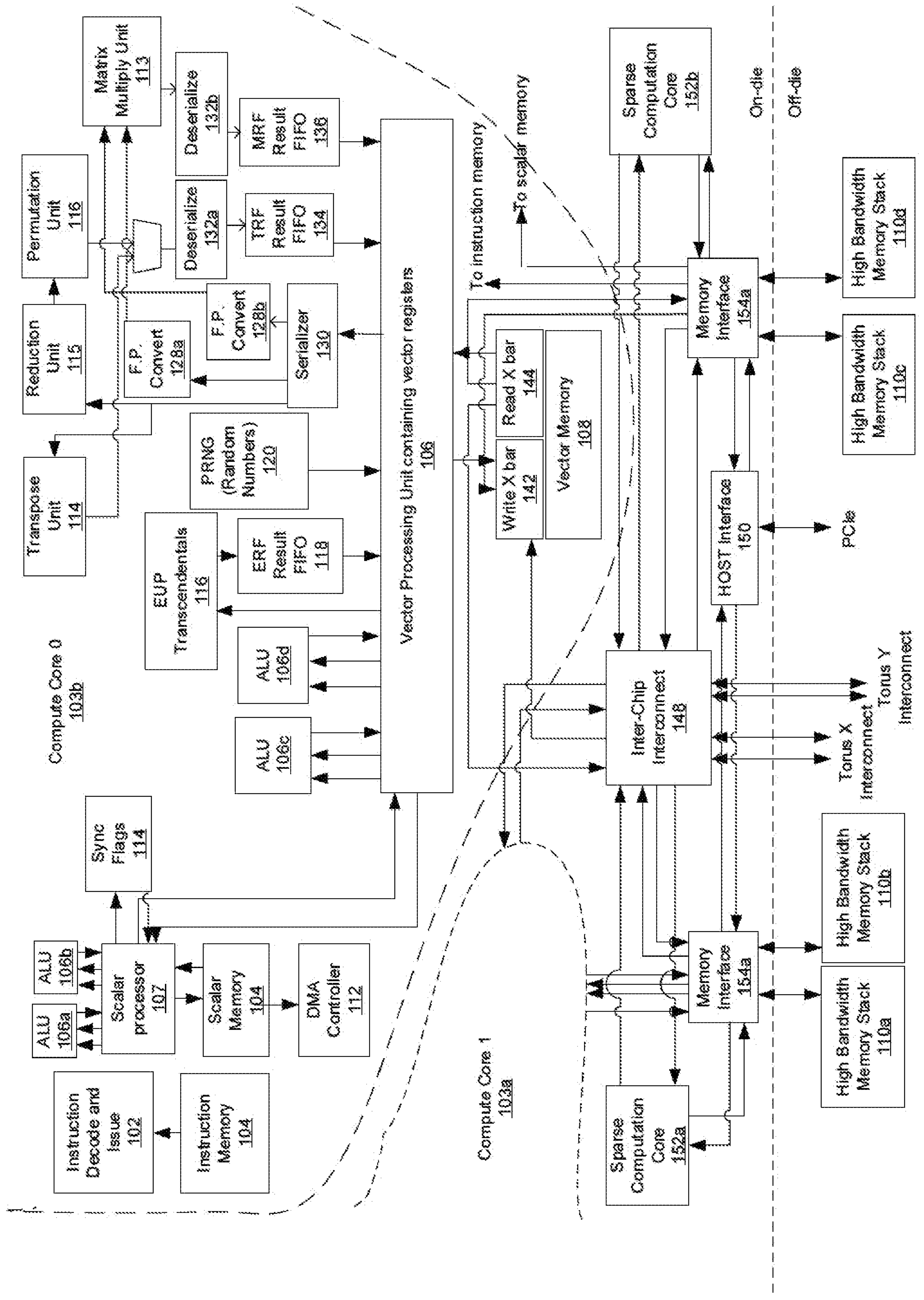
Compute Core 0 103b
Instruction Decode and Issue 102
Instruction Memory 104
ALU 106a
ALU 106b
Scalar processor 107
Scalar Memory 104
DMA Controller 112
Sync Flags 114
Transpose Unit 114
Reduction Unit 115
Permutation Unit 116
Matrix Multiply Unit 113
F.P. Convert 128a
F.P. Convert 128b
Deserialize 132a
Deserialize 132b
Serializer 130
TRF Result FIFO 134
MRF Result FIFO 136
PRNG (Random Numbers) 120
EUP Transcendentals 116
ERF Result FIFO 118
ALU 106c
ALU 106d
Vector Processing Unit containing vector registers 106
To instruction memory
To scalar memory
Read X bar 144
Write X bar 142
Vector Memory 108
Inter-Chip Interconnect 148
Sparse Computation Core 152b
Sparse Computation Core 152a
Memory Interface 154a
Memory Interface 154a
HOST Interface 150
PCIe
Torus X Interconnect
Torus Y Interconnect
On-die
Off-die
High Bandwidth Memory Stack 110d
High Bandwidth Memory Stack 110c
High Bandwidth Memory Stack 110b
High Bandwidth Memory Stack 110a
Compute Core 1 103a

FIG. 1C

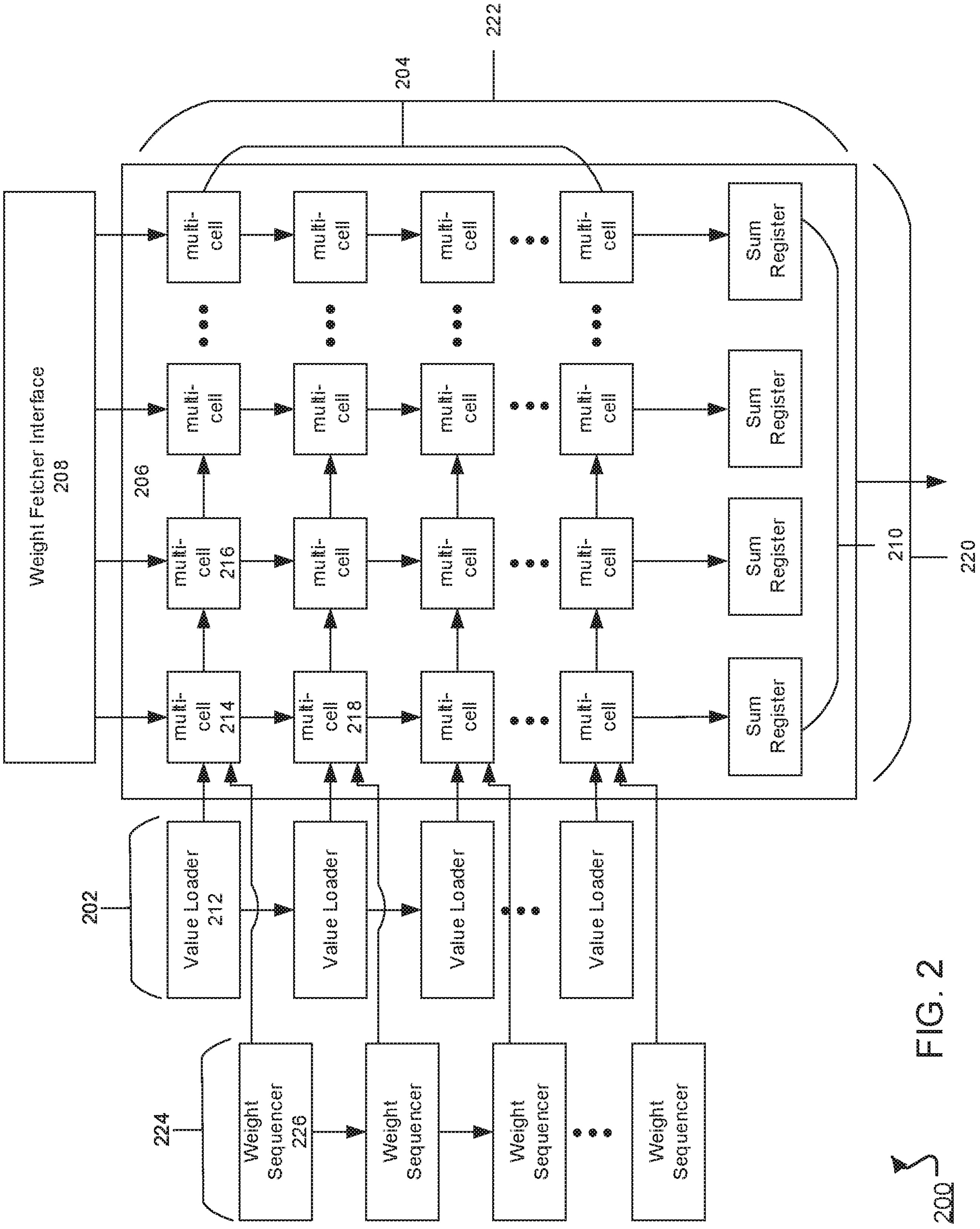
Weight Fetcher Interface
208
206
204
222
multi-cell
multi-cell 214
multi-cell 216
multi-cell 218
Sum Register
210
220
Value Loader 212
Value Loader
202
Weight Sequencer 226
Weight Sequencer
224
200

FIG. 2

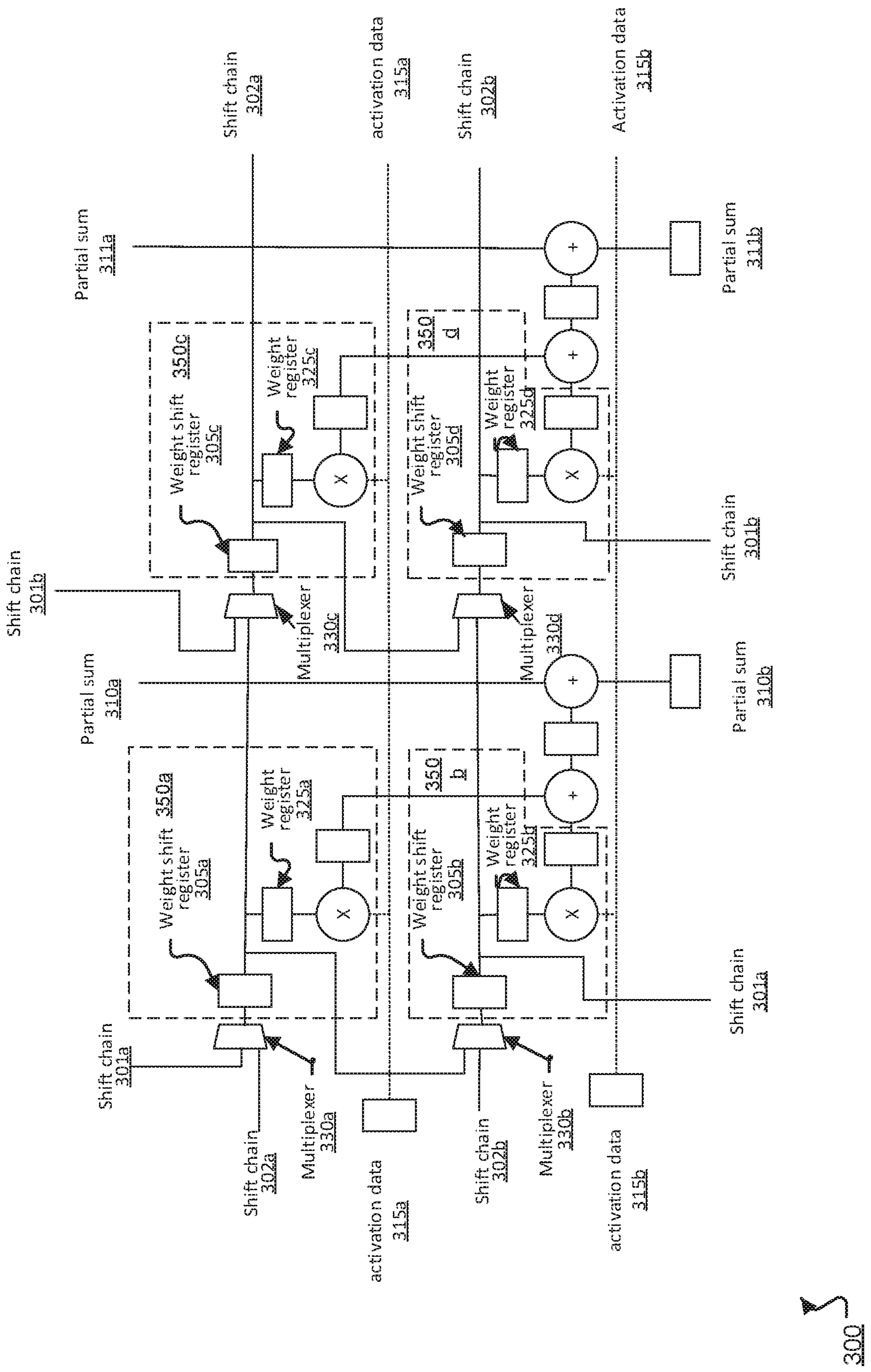
Shift chain 302a
activation data 315a
Shift chain 302b
Activation data 315b
Partial sum 311a
Partial sum 311b
Weight shift register 305c
350c
Weight register 325c
350 d
Weight shift register 305d
Weight register 325d
Shift chain 301b
Multiplexer 330c
Multiplexer 330d
Partial sum 310a
Partial sum 310b
Weight shift register 305a
350a
Weight register 325a
350 b
Weight shift register 305b
Weight register 325b
Shift chain 301a
Multiplexer 330a
Multiplexer 330b
activation data 315b
300

FIG. 3

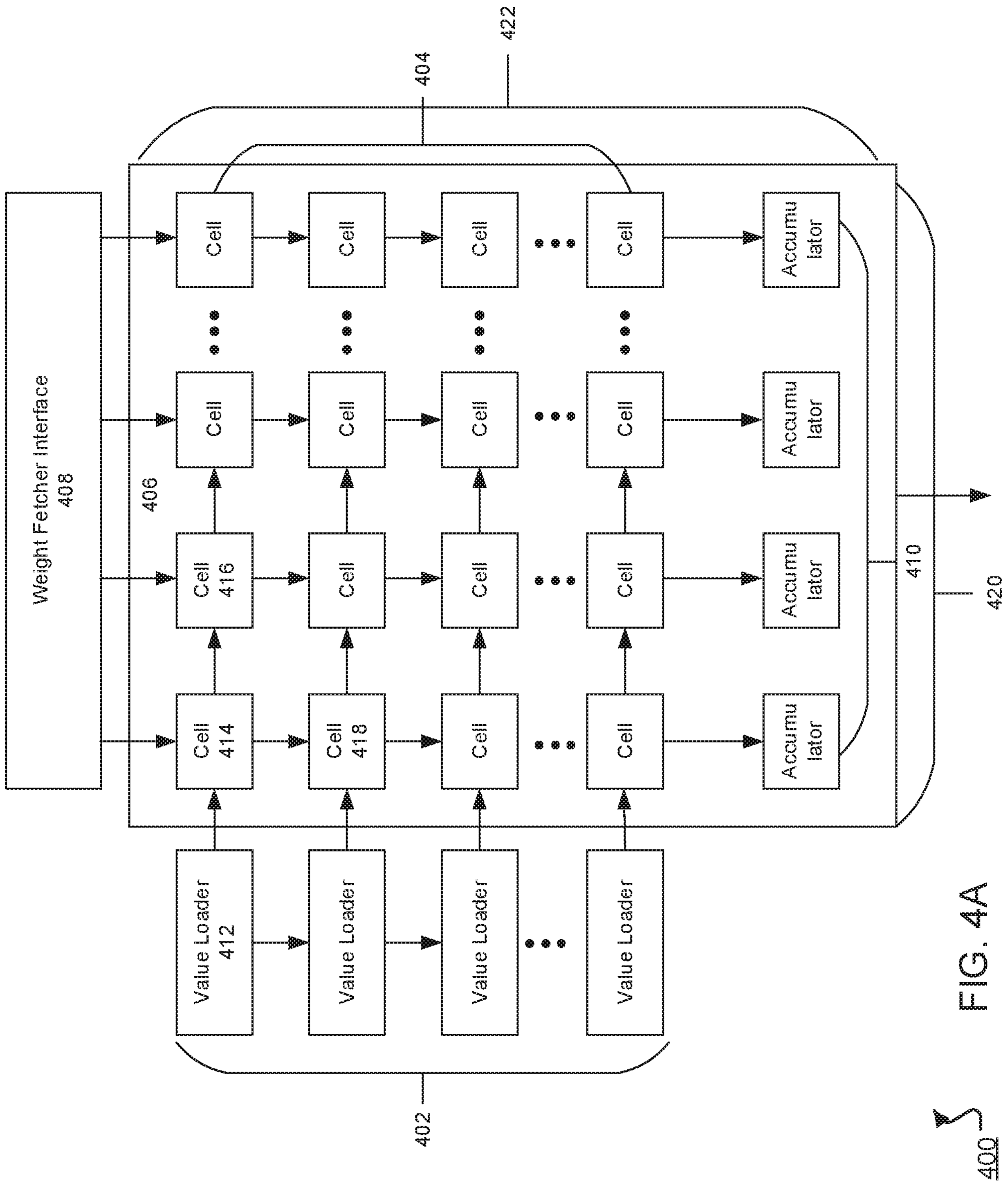
Weight Fetcher Interface 408
406
404
422
Cell 414
Cell 416
Cell 418
Cell
Accumu lator
Value Loader 412
Value Loader
402
410
420
400

FIG. 4A

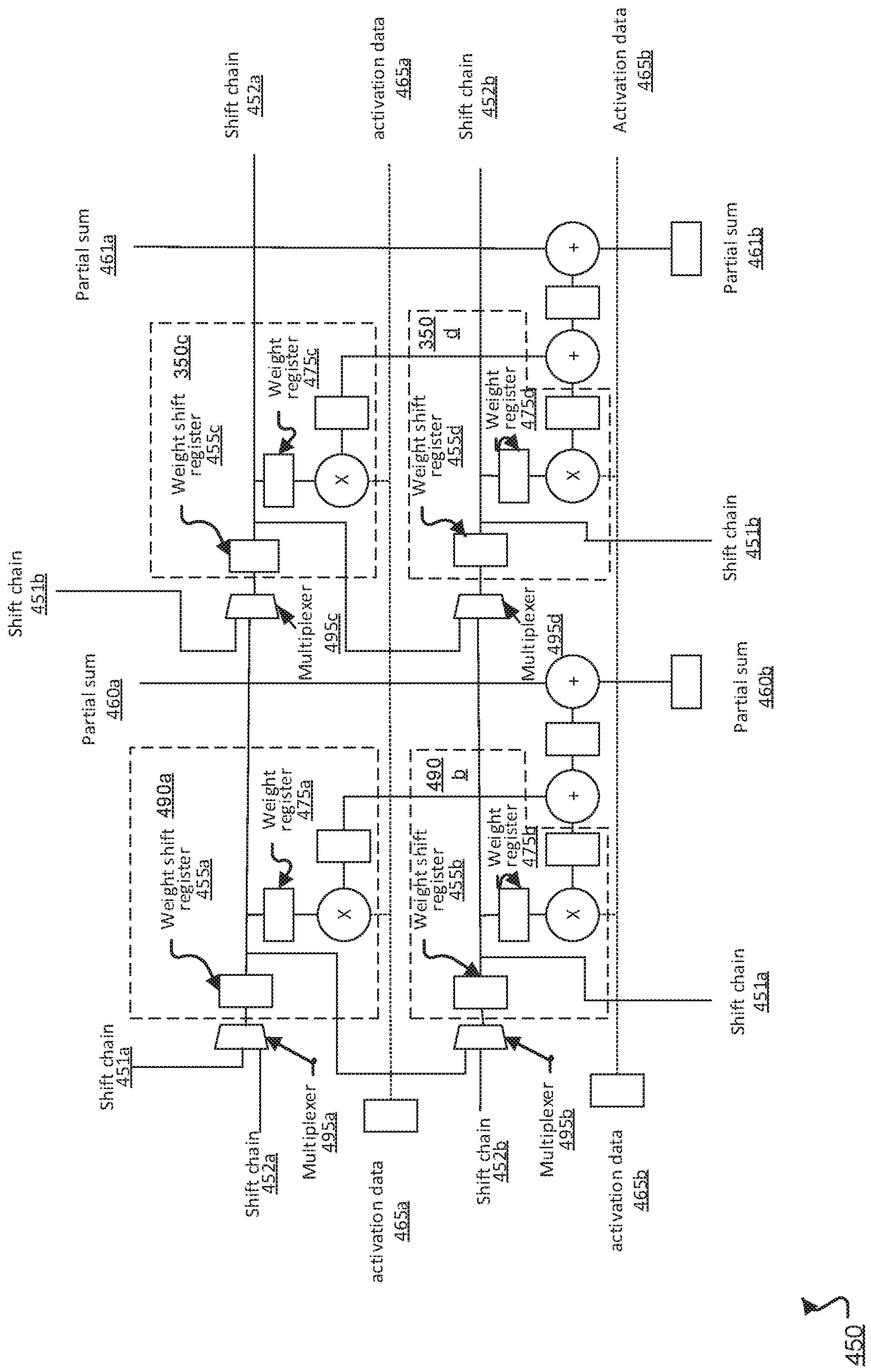
Shift chain 452a
activation data 465a
Shift chain 452b
Activation data 465b
Partial sum 461a
Partial sum 461b
Weight shift register 455c
350c
Weight register 475c
350 d
Weight shift register 455d
Weight register 475d
Shift chain 451b
Shift chain 451b
Multiplexer 495c
Multiplexer 495d
Partial sum 460a
Partial sum 460b
Weight shift register 455a
490a
Weight register 475a
490 b
Weight shift register 455b
Weight register 475b
Shift chain 451a
Shift chain 451a
Shift chain 452a
Multiplexer 495a
activation data 465a
Shift chain 452b
Multiplexer 495b
activation data 465b
450

FIG. 4B

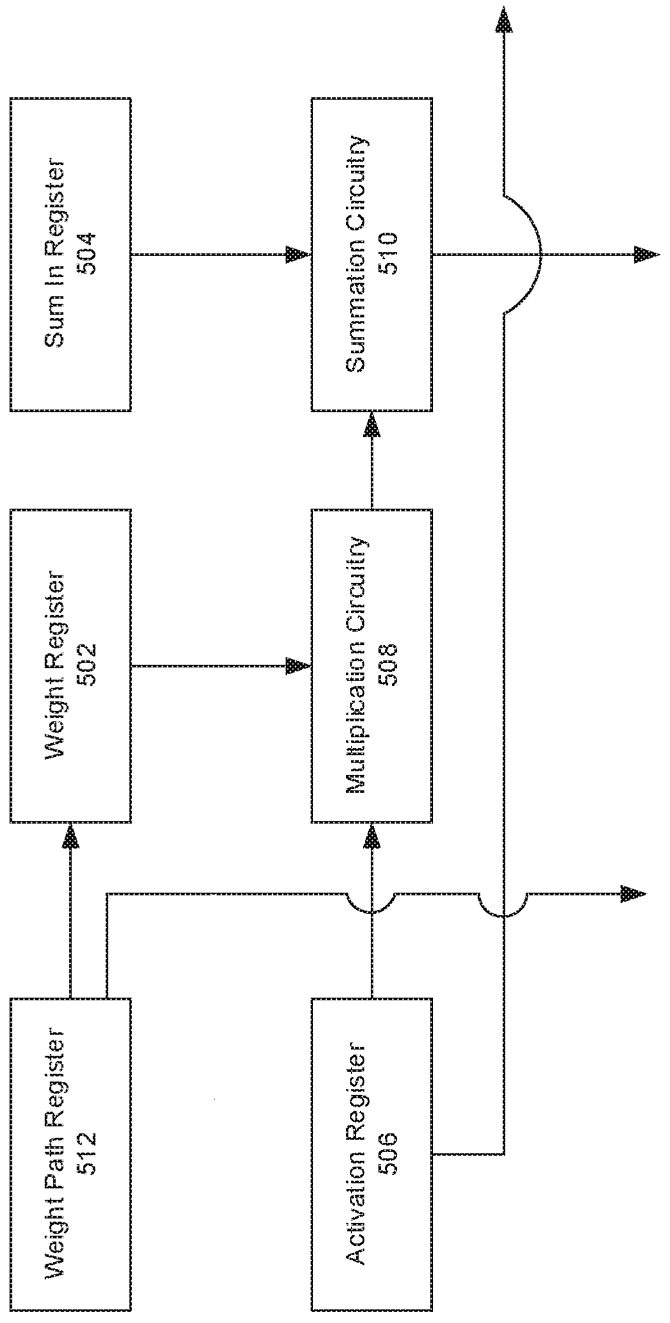
Sum In Register 504
Summation Circuitry 510
Weight Register 502
Multiplication Circuitry 508
Weight Path Register 512
Activation Register 506

FIG. 5

500

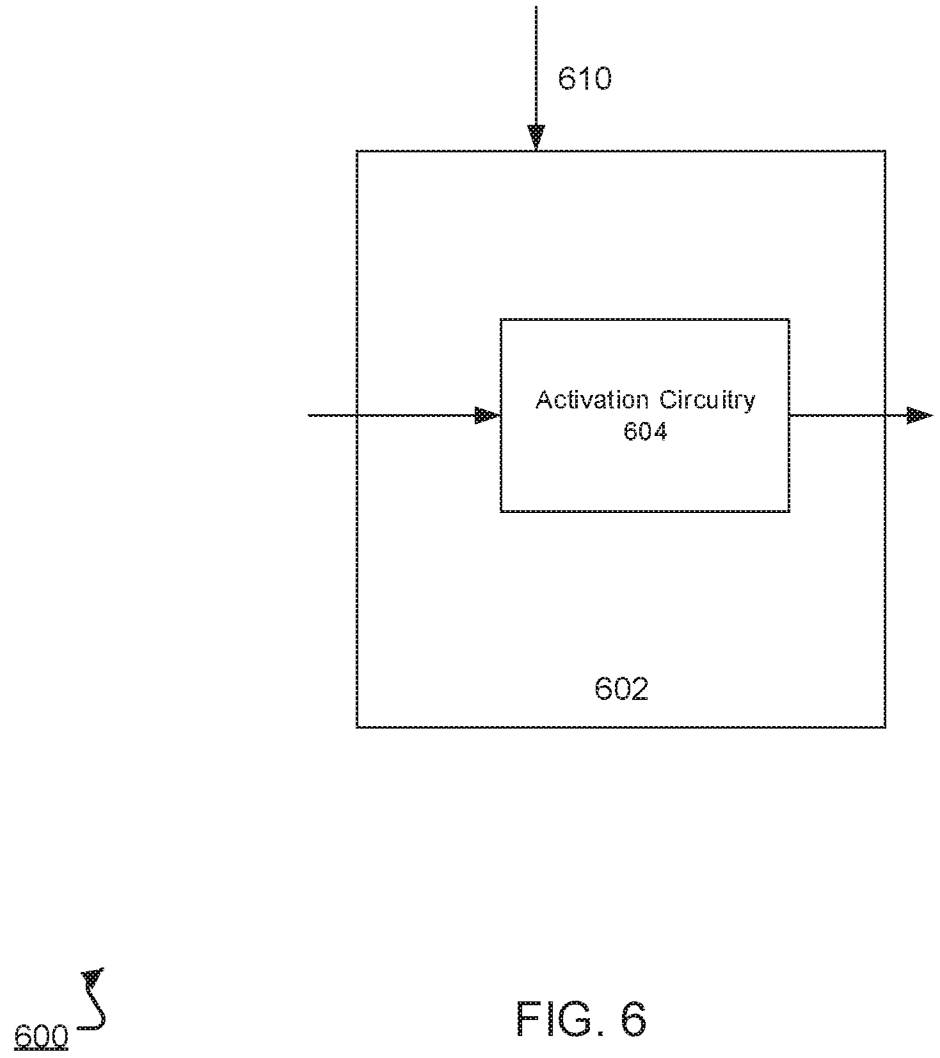

FIG. 6

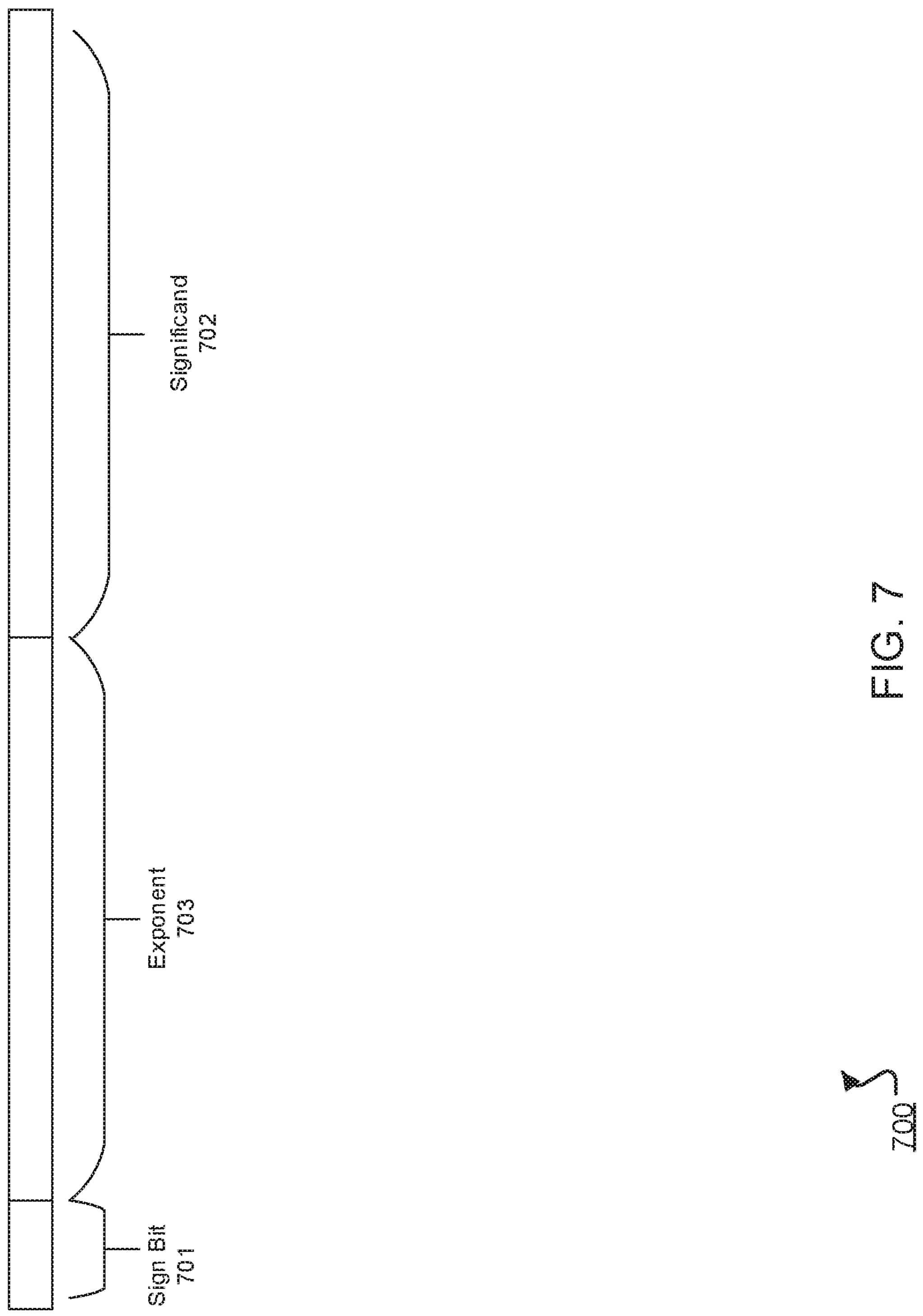

FIG. 7

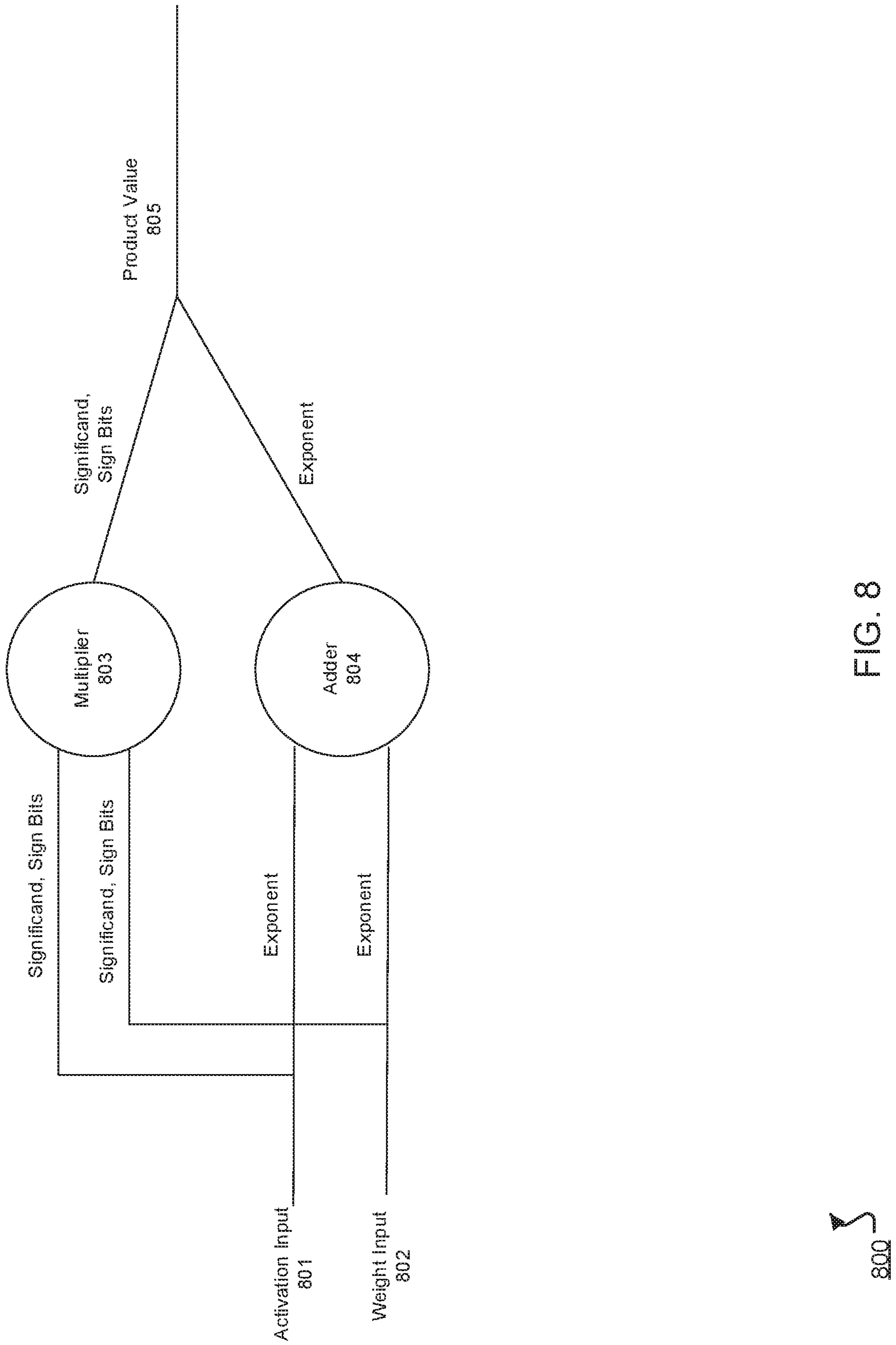
Product Value 805
Significand, Sign Bits
Exponent
Multiplier 803
Adder 804
Significand, Sign Bits
Significand, Sign Bits
Exponent
Exponent
Activation Input 801
Weight Input 802
800

FIG. 8

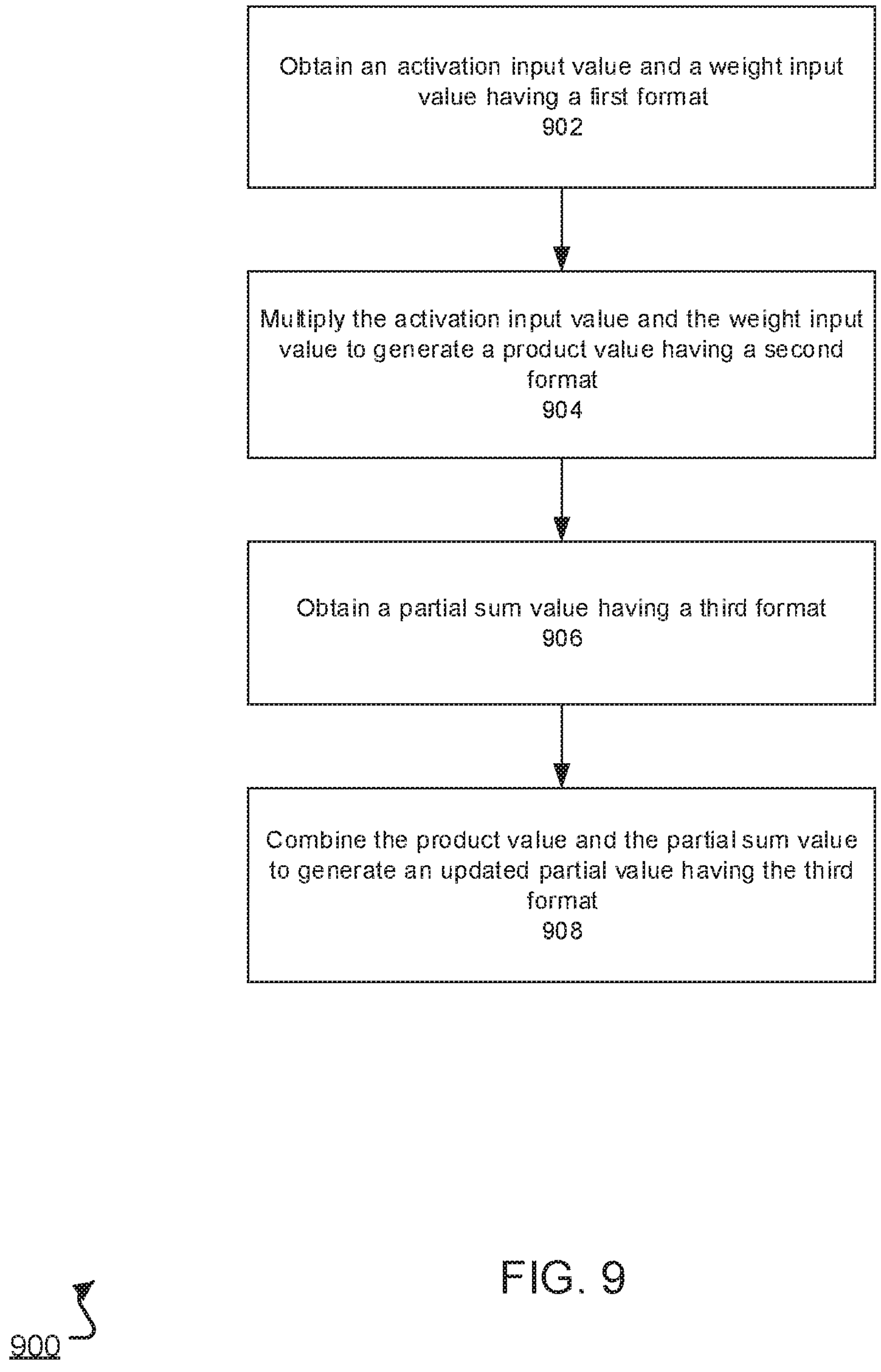
Obtain an activation input value and a weight input value having a first format
902
Multiply the activation input value and the weight input value to generate a product value having a second format
904
Obtain a partial sum value having a third format
906
Combine the product value and the partial sum value to generate an updated partial value having the third format
908
900

FIG. 9

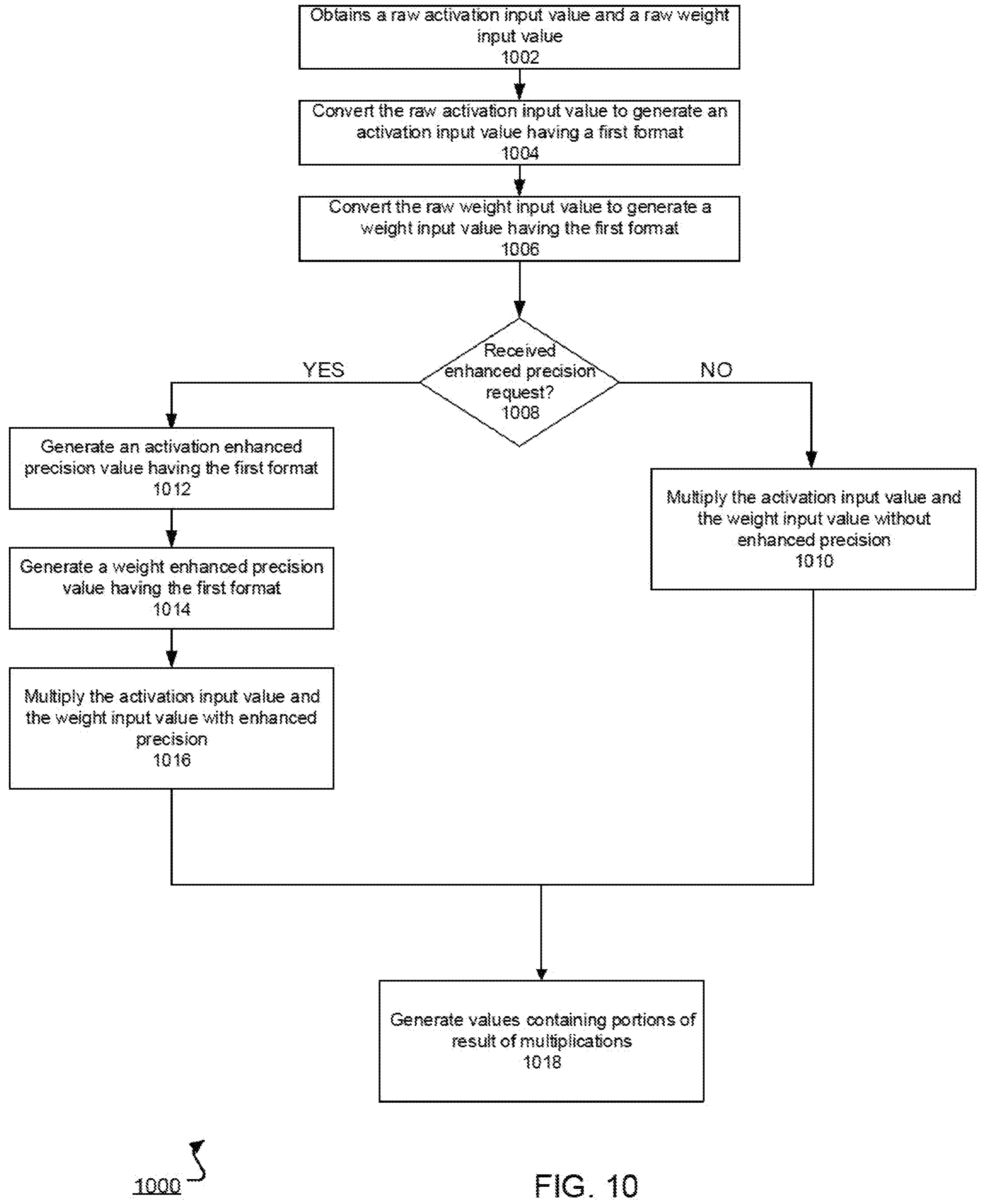
Obtains a raw activation input value and a raw weight input value
1002
Convert the raw activation input value to generate an activation input value having a first format
1004
Convert the raw weight input value to generate a weight input value having the first format
1006
Received enhanced precision request?
1008
YES
NO
Generate an activation enhanced precision value having the first format
1012
Generate a weight enhanced precision value having the first format
1014
Multiply the activation input value and the weight input value with enhanced precision
1016
Multiply the activation input value and the weight input value without enhanced precision
1010
Generate values containing portions of result of multiplications
1018
1000

FIG. 10

Electronic Patent Application Fee Transmittal

Application Number:	
Filing Date:	
Title of Invention:	PERFORMING MATRIX MULTIPLICATION IN HARDWARE
First Named Inventor/Applicant Name:	Andrew Everett Phelps
Filer:	STEPHANIE D. GROSVENOR/Christine Rogers
Attorney Docket Number:	16113-8252P01

Filed as Large Entity

Filing Fees for **Provisional**

Description	Fee Code	Quantity	Amount	Sub-Total in USD($)
Basic Filing:				
PROVISIONAL APPLICATION FILING	1005	1	260	260
Pages:				
Claims:				
Miscellaneous-Filing:				
Petition:				
Patent-Appeals-and-Interference:				
Post-Allowance-and-Post-Issuance:				

Description	Fee Code	Quantity	Amount	Sub-Total in USD($)
Extension-of-Time:				
Miscellaneous:				
	Total in USD ($)			**260**

Electronic Acknowledgement Receipt

EFS ID:	29242474
Application Number:	62507748
International Application Number:	
Confirmation Number:	3256
Title of Invention:	PERFORMING MATRIX MULTIPLICATION IN HARDWARE
First Named Inventor/Applicant Name:	Andrew Everett Phelps
Customer Number:	26192
Filer:	STEPHANIE D. GROSVENOR/Denise Siede
Filer Authorized By:	STEPHANIE D. GROSVENOR
Attorney Docket Number:	16113-8252P01
Receipt Date:	17-MAY-2017
Filing Date:	
Time Stamp:	21:40:48
Application Type:	Provisional

Payment information:

Submitted with Payment	yes
Payment Type	DA
Payment was successfully received in RAM	$260
RAM confirmation Number	051817INTEFSW00007564061050
Deposit Account	
Authorized User	

The Director of the USPTO is hereby authorized to charge indicated fees and credit any overpayment as follows:

File Listing:

Document Number	Document Description	File Name	File Size(Bytes)/ Message Digest	Multi Part /.zip	Pages (if appl.)
1	Provisional Cover Sheet (SB16)	16113_8252P01_Transmittal.pdf	125557 8df3e9a9b6bdabe9a2abc91d02ff86ae50b4b908	no	1

Warnings:

This is not a USPTO supplied Provisional Cover Sheet SB16 form.

Information:

Document Number	Document Description	File Name	File Size(Bytes)/ Message Digest	Multi Part /.zip	Pages (if appl.)
2		16113_8252P01_Specification.pdf	211132 c7535f024bef7fc8d8b9f3a7d85bcf4f5bf8b786	yes	39

Multipart Description/PDF files in .zip description

Document Description	Start	End
Specification	1	34
Claims	35	39

Warnings:

Information:

Document Number	Document Description	File Name	File Size(Bytes)/ Message Digest	Multi Part /.zip	Pages (if appl.)
3	Application Data Sheet	16113_8252P01_ADS.pdf	1822847 09a2b0f06babcdfe79a417e0c66d7684989a8041	no	8

Warnings:

Information:

Document Number	Document Description	File Name	File Size(Bytes)/ Message Digest	Multi Part /.zip	Pages (if appl.)
4	Drawings-only black and white line drawings	16113_8252P01_Drawings.pdf	745493 fd1a69cafe80a2f78978deb89574eef3f0f6f36c	no	13

Warnings:

Information:

Document Number	Document Description	File Name	File Size(Bytes)/ Message Digest	Multi Part /.zip	Pages (if appl.)
5	Fee Worksheet (SB06)	fee-info.pdf	30017 1b4544760ff614bce178a44df0d386b8dda13841	no	2

Warnings:

Information:	
Total Files Size (in bytes):	2935046

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

New Applications Under 35 U.S.C. 111

If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.

National Stage of an International Application under 35 U.S.C. 371

If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.

New International Application Filed with the USPTO as a Receiving Office

If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NUMBER	FILING or 371(c) DATE	GRP ART UNIT	FIL FEE REC'D	ATTY.DOCKET.NO	TOT CLAIMS	IND CLAIMS
62/507,748	05/17/2017		260	16113-8252P01		

CONFIRMATION NO. 3256

FILING RECEIPT

OC000000091653425

26192
FISH & RICHARDSON P.C.
PO BOX 1022
MINNEAPOLIS, MN 55440-1022

Date Mailed: 05/26/2017

Receipt is acknowledged of this provisional patent application. It will not be examined for patentability and will become abandoned not later than twelve months after its filing date. Any correspondence concerning the application must include the following identification information: the U.S. APPLICATION NUMBER, FILING DATE, NAME OF APPLICANT, and TITLE OF INVENTION. Fees transmitted by check or draft are subject to collection. Please verify the accuracy of the data presented on this receipt. **If an error is noted on this Filing Receipt, please submit a written request for a Filing Receipt Correction. Please provide a copy of this Filing Receipt with the changes noted thereon. If you received a "Notice to File Missing Parts" for this application, please submit any corrections to this Filing Receipt with your reply to the Notice. When the USPTO processes the reply to the Notice, the USPTO will generate another Filing Receipt incorporating the requested corrections**

Inventor(s)

Andrew Everett Phelps, Middleton, WI;
Norman Paul Jouppi, Palo Alto, CA;

Applicant(s)

Google Inc., Mountain View, CA

Power of Attorney:
Stephanie Grosvenor--67592

Permission to Access Application via Priority Document Exchange: Yes

Permission to Access Search Results: Yes

Applicant may provide or rescind an authorization for access using Form PTO/SB/39 or Form PTO/SB/69 as appropriate.

If Required, Foreign Filing License Granted: 05/25/2017
The country code and number of your priority application, to be used for filing abroad under the Paris Convention, is **US 62/507,748**
Projected Publication Date: None, application is not eligible for pre-grant publication
Non-Publication Request: No
Early Publication Request: No
Title

PERFORMING MATRIX MULTIPLICATION IN HARDWARE

Statement under 37 CFR 1.55 or 1.78 for AIA (First Inventor to File) Transition Applications: No

PROTECTING YOUR INVENTION OUTSIDE THE UNITED STATES

Since the rights granted by a U.S. patent extend only throughout the territory of the United States and have no effect in a foreign country, an inventor who wishes patent protection in another country must apply for a patent in a specific country or in regional patent offices. Applicants may wish to consider the filing of an international application under the Patent Cooperation Treaty (PCT). An international (PCT) application generally has the same effect as a regular national patent application in each PCT-member country. The PCT process **simplifies** the filing of patent applications on the same invention in member countries, but **does not result** in a grant of "an international patent" and does not eliminate the need of applicants to file additional documents and fees in countries where patent protection is desired.

Almost every country has its own patent law, and a person desiring a patent in a particular country must make an application for patent in that country in accordance with its particular laws. Since the laws of many countries differ in various respects from the patent law of the United States, applicants are advised to seek guidance from specific foreign countries to ensure that patent rights are not lost prematurely.

Applicants also are advised that in the case of inventions made in the United States, the Director of the USPTO must issue a license before applicants can apply for a patent in a foreign country. The filing of a U.S. patent application serves as a request for a foreign filing license. The application's filing receipt contains further information and guidance as to the status of applicant's license for foreign filing.

Applicants may wish to consult the USPTO booklet, "General Information Concerning Patents" (specifically, the section entitled "Treaties and Foreign Patents") for more information on timeframes and deadlines for filing foreign patent applications. The guide is available either by contacting the USPTO Contact Center at 800-786-9199, or it can be viewed on the USPTO website at http://www.uspto.gov/web/offices/pac/doc/general/index.html.

For information on preventing theft of your intellectual property (patents, trademarks and copyrights), you may wish to consult the U.S. Government website, http://www.stopfakes.gov. Part of a Department of Commerce initiative, this website includes self-help "toolkits" giving innovators guidance on how to protect intellectual property in specific countries such as China, Korea and Mexico. For questions regarding patent enforcement issues, applicants may call the U.S. Government hotline at 1-866-999-HALT (1-866-999-4258).

LICENSE FOR FOREIGN FILING UNDER

Title 35, United States Code, Section 184

Title 37, Code of Federal Regulations, 5.11 & 5.15

GRANTED

The applicant has been granted a license under 35 U.S.C. 184, if the phrase "IF REQUIRED, FOREIGN FILING LICENSE GRANTED" followed by a date appears on this form. Such licenses are issued in all applications where the conditions for issuance of a license have been met, regardless of whether or not a license may be required as set forth in 37 CFR 5.15. The scope and limitations of this license are set forth in 37 CFR 5.15(a) unless an earlier license has been issued under 37 CFR 5.15(b). The license is subject to revocation upon written notification. The date indicated is the effective date of the license, unless an earlier license of similar scope has been granted under 37 CFR 5.13 or 5.14.

This license is to be retained by the licensee and may be used at any time on or after the effective date thereof unless it is revoked. This license is automatically transferred to any related applications(s) filed under 37 CFR 1.53(d). This license is not retroactive.

The grant of a license does not in any way lessen the responsibility of a licensee for the security of the subject matter as imposed by any Government contract or the provisions of existing laws relating to espionage and the national security or the export of technical data. Licensees should apprise themselves of current regulations especially with respect to certain countries, of other agencies, particularly the Office of Defense Trade Controls, Department of State (with respect to Arms, Munitions and Implements of War (22 CFR 121-128)); the Bureau of Industry and Security, Department of Commerce (15 CFR parts 730-774); the Office of Foreign AssetsControl, Department of Treasury (31 CFR Parts 500+) and the Department of Energy.

NOT GRANTED

No license under 35 U.S.C. 184 has been granted at this time, if the phrase "IF REQUIRED, FOREIGN FILING LICENSE GRANTED" DOES NOT appear on this form. Applicant may still petition for a license under 37 CFR 5.12, if a license is desired before the expiration of 6 months from the filing date of the application. If 6 months has lapsed from the filing date of this application and the licensee has not received any indication of a secrecy order under 35 U.S.C. 181, the licensee may foreign file the application pursuant to 37 CFR 5.15(b).
